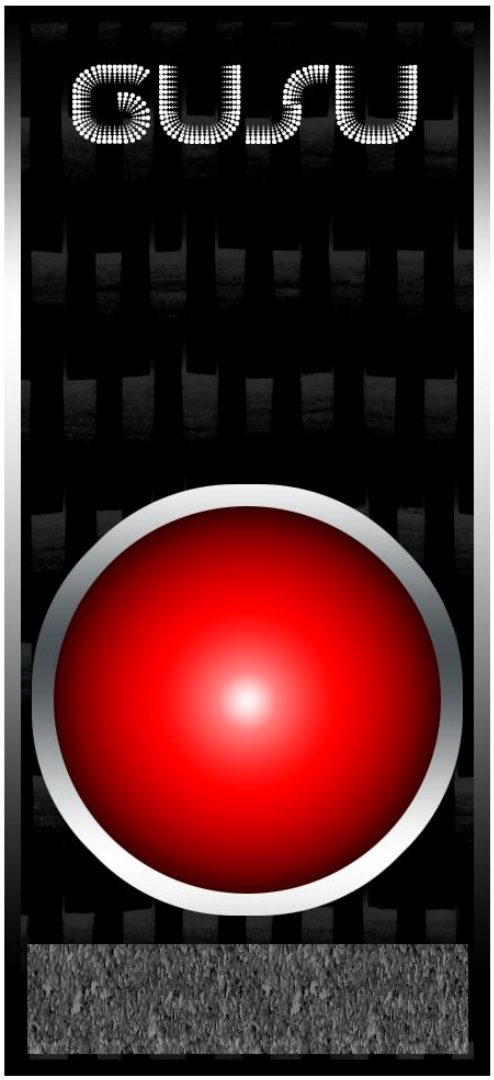


Get Up
Stand Up

It knows when you're sleeping
It knows when you're awake

EEL4914 | April 27, 2009 | Group 5

Philip Bell
Andrew Léger
Matthew O'Morrow
Joshua Rust

Table of Contents

1.0 Executive Summary ..... - 1 -

1.1 Problem Statement ..... - 2 -

1.2 Solution ..... - 3 -

1.3 Research Methodology ..... - 4 -

1.4 Design Methodology ..... - 4 -

1.5 Implementation Methodology ..... - 5 -

1.6 Project Management ..... - 6 -

1.7 Similar Projects ..... - 8 -

2.0 Case Design ..... - 10 -

2.1 ABS Plastic Enclosures ..... - 11 -

2.2 Lucite Sheets ..... - 11 -

2.3 Wood ..... - 12 -

3.0 Microcontroller ..... - 13 -

3.1 Technical Objectives and Specifications ..... - 13 -

3.1.1 Goals ..... - 13 -

3.2 Research ..... - 14 -

3.2.1 Texas Instruments MSP430 ..... - 15 -

3.2.2 Atmel AVR ..... - 15 -

3.3 Implementation ..... - 17 -

3.4 Test Plan ..... - 18 -

4.0 Alarm Module ..... - 19 -

4.1 Block Diagram ..... - 19 -

4.2. Design Requirements ..... - 20 -

4.2.1 Audio Output ..... - 20 -

4.2.2 FM Tuner ..... - 25 -

4.2.3 MP3 Decoder ..... - 32 -

4.2.4 Sim Card Reader ..... - 38 -

5.0 User Interface ..... - 45 -

5.1 Motivation ..... - 45 -

5.2 Liquid Crystal Display ..... - 46 -

5.2.1 Research ..... - 46 -

5.2.2 Implementation Strategy ..... - 50 -

5.2.3 LCD Driver Chip ..... - 52 -

5.2.4 Test Plan ..... - 53 -
5.3 Physical User Interface ..... - 54 -
5.3.1 User's Expectation ..... - 54 -
5.3.2 Key switch ..... - 56 -
5.3.3 Physical User Interface Implementation ..... - 57 -
5.3.4 Physical User Interface Test Plan ..... - 58 -
5.4 Graphical User Interface ..... - 58 -
5.5 Complete User Interface ..... - 60 -
6.0 Sensor System ..... - 61 -
6.0.1 Motivation ..... - 61 -
6.0.2 Assumptions ..... - 61 -
6.1 Research ..... - 62 -
6.1.1 Weight Sensing ..... - 62 -
6.1.2 Distance Sensing ..... - 65 -
6.1.3 Infrared ..... - 66 -
6.2 Implementation ..... - 68 -
6.3: Sensor System Testing ..... - 70 -
7.0 Radio Frequency Transmission ..... - 70 -
7.0.1 Design and Requirements ..... - 70 -
7.0.2 Research ..... - 71 -
7.0.3 Implementation ..... - 76 -
7.0.4 Configuration ..... - 77 -
7.0.5 Test Plan ..... - 78 -
7.1 Remote Detection Unit ..... - 79 -
7.1.1 Technical Objectives and Specifications ..... - 79 -
7.1.2 Goals ..... - 79 -
7.1.3 Input/output ..... - 79 -
7.1.4 Power ..... - 79 -
7.1.5 Software ..... - 79 -
7.1.6 Research ..... - 80 -
7.1.7 Implementation ..... - 80 -
7.1.8 Test Plan ..... - 82 -
7.2 Coffee Machine ..... - 82 -
7.2.1 Technical Objectives and Specifications ..... - 82 -
7.2.2 Research ..... - 84 -

7.2.3 Implementation ..... - 85 -
7.2.4 Test Plan ..... - 87 -
8.0 Clock Module ..... - 87 -
8.1 Goals and Objectives ..... - 87 -
8.2 Research ..... - 87 -
8.2.1 Atomic Clock ..... - 87 -
8.2.2 Real Time Clock ..... - 88 -
8.3 Implementation ..... - 91 -
8.4 Test Plan ..... - 91 -
9.0 Power Supply ..... - 91 -
9.1 Block Diagram ..... - 91 -
9.2 Power System ..... - 92 -
9.2.1 Power Supply ..... - 92 -
9.2.2 Battery Backup ..... - 102 -
10.0 Software ..... - 104 -
10.0.1 Motivation ..... - 104 -
10.1 Software Research ..... - 104 -
10.1.1 Alternatives ..... - 104 -
10.2 Software Implementation ..... - 106 -
10.2.1 General Configuration ..... - 106 -
10.2.2 Overall Behavior ..... - 109 -
10.2.3 Functions ..... - 112 -
10.3 Software Testing ..... - 113 -
11.0 Printed Circuit Board ..... - 114 -
11.1 Research ..... - 114 -
11.2 Implementation ..... - 116 -
12.0 Budget ..... - 118 -
13.0 Final Design Review & Integration Test ..... - 120 -
14.0 Conclusion ..... - 123 -
APPENDIX A: Works Cited ..... - 124 -
APPENDIX B: Emails ..... - 128 -

1.0 Executive Summary

There is always a need for alarm clocks in our daily society. They are used for many reasons. The most common and popular uses are to be woken up in the morning for school or work. Although these would seem the obvious reasons there are others such as reminding the user about a certain event in their day or even to let them know when a deadline needs to be completed. Whatever the reason is, there is always a demand for these ingenious and effective electronic devices.

With the growing technical revolution taking place, electronic companies are continually researching new ways to improve upon the design of current alarm clocks. Some of these new modifications include an improved visual screen, better sound quality, and more features that can be used in the common person's lifestyle. More and more companies are using LCD screen displays as an aesthetic way to illustrate the software and features that these new alarm clocks have.

While most alarm clocks contain an AM/FM tuner, some corporations are looking into satellite radio to incorporate into their designs. Using this technology would bring in a much broader audience than the regular AM/FM users because the general public would be exposed to other stations than just AM/FM. Sound quality is something people are always looking to improve upon and developing new types of sound speakers are high on the list of improvements made to the daily alarm clocks. New software is always being developed to improve the sound quality of the actual music files that can be played on the newer alarm clocks such as MP3, FLAC, and OGG. To use this improved software and file formats new alarm clocks must have software developed to meet these requirements.

One other feature that has been gaining remarkable popularity recently has been the addition of an Ethernet jack or wireless capability so the user can connect their alarm clock into their home network. The major advantages of this would be so that the home user can awake to the local news, get weather reports from their area, and even to check the regional traffic reports so that they can adjust their commute to work or school accordingly. This new technology has been gaining momentum with the developing companies as to provide software and firmware upgrades to their alarm clocks to keep their systems up to date. New updates have been made available to these alarm clocks such as new user interfaces and downloadable content.

The "Go Go" American lifestyle has changed throughout recent decades with more and more people working longer hours. In consequence to this, more people are losing sleep due to the constant stress of work, school, and children. But in order to keep their current lifestyles these demands must be met. Figure

1.0.1 shows a day in the life of a typical American employee as researched by the National Sleep Foundation.

Figure 1.0.1: Shows the day in the life of a typical American Worker. Reprinted with permission from the National Sleep Foundation.

According to the National Sleep Foundation the average time spent sleeping every night is between six and seven hours, while the National Sleep Foundation recommends that people get between seven and nine hours of sleep to function at their peak performance. Some other staggering statistics that were found is that about 25% of American workers tend to take a quick nap while at the workplace and more than half of the American workforce brings home some of their work to complete [1]. These figures show how not getting enough sleep at home can hinder employment productivity and performance.

Now, bear in mind that most alarm clocks cannot solve this dilemma but they do prove to be useful in dealing with this daily occurrence. Modern alarm clocks allow people to prioritize their time more efficiently. In today's progressive society time is of the essence and in the business world time is money. With today's alarm clocks they help give people's lives a more structured schedule, this can help them in having a better routine which in turn can help them get a better night's rest. Current alarm clocks have lost their effectiveness in dealing with these sleep problems. The GuSu prototype will attempt to solve these issues with a better design that will help people with their dynamic lifestyles.

1.1 Problem Statement

Waking up, especially for some college students, can be troublesome. In a worst case scenario an individual may have a half dozen alarms scattered throughout a room set for one or many times, perhaps even with a few outside their room, and still find themselves shutting all of these alarms off and getting back in bed, only to wake up later realizing they are late to start the day. Most solutions proposed and even commercially available have the alarm clock "launch" an object that must be retrieved, or task the individual with solving some form of problem or puzzle to quiet the alarm. However this does not defeat the individual who is able to leave their room to shut off an alarm and still get back in bed, because in essence this still leaves the person with a snooze or turn off method for the alarm, and perhaps after a few weeks will become very adept at whatever task

must be performed to silence the alarm, and now have not only spent time dealing with the alarm, but also gone back to bed, thus resulting in even more delay to their morning routine.

An alarm clock serves three purposes; waking the user when they wish to be awoken, getting the user out of bed, and waking them in a "non-harmful" or disorientating way. These three requirements can be further broken down in the following way. The alarm clock obviously must be able to keep track of time, and let the user decide what time they want to wake up. Waking up the user alone is not enough, the ideal alarm clock ensures that the user also gets up and out of bed in a timely fashion and stays out of bed (by having no snooze or off button), in essence preventing both oversleeping and snoozing. Finally, users have different needs and preferences for being roused from slumber. For example, some individuals require sudden, often loud or jolting methods, while others wake better to a slow process, such as subtle music or talk radio followed by lights coming on, scents of coffee, or an old fashioned alarm tone signifying the time to get up has passed.

1.2 Solution

Building a clock from the ground up that has no snooze or off button and removing any direct action on part of the user that can silence the alarm is the first step in satisfying the three aforementioned design necessities of a better alarm clock system. But the alarm must turn off somehow, so the clock will include sensors that tell it whether someone is in the bed or not, and will run off of the simple logic that if it is time to get up, and someone is in the bed, it will follow it's programmed routine. This motivates the user by giving them no option but to get out of bed and stay out of bed. In addition, a battery backup system serves two purposes; it helps keep the internal clock correct, and ensures the user cannot silence the alarm by unplugging its power source. The internal clock itself may be either an atomic clock, set for time-zone by the user, or a binary clock adjusted by the user.

For meeting a user's "waking preferences" the clock shall have the following abilities that tie in to the alarm clocks wireless modules and alarm speaker. An FM tuner, an SD card slot for mp3 storage and mp3 decoder chip, a built in common tone alarm, Zigbee interfaces with lights and coffee makers, seven day programmability, and timing/ordering adjustments for what alarm option will occur (for example if the user would like the FM radio to play through the speaker fifteen minutes before the wake up time, at which point lights may turn on and the alarm tone will sound).

To further prevent the user "tricking" the clock into silencing, the alarm settings can only be changed and tested when a turnkey slot on the bottom of the unit is turned from the Running mode to a Test/Set mode. This turnkey will be ignored during the alarm period.

1.3 Research Methodology

The purpose of research methods is to successfully explore information that is relevant to the design that will be put into practice. There are various methods of research that are used all the time throughout the academic and professional disciplines. Some of these methods include use of a library, the internet, and even life experiences. The purpose of doing research before designing a project is to measure the idea's originality and creativity. There are other factors that will be taken into account including the significance, contribution, and technical soundness of the design at hand. Looking at research from another point of view there are two main aspects that define research; those are qualitative versus quantitative. When comparing both of these measures of research the qualitative side can be defined as the measure of the quality of the project. This means that the qualitative measures will be found in the research and analysis of the data and physical attributes of the design. On the other side there is the quantitative approach which includes the research of and analysis of the numerical data. This side focuses more on the numbers and equations that will hold the design together.

The main method that will be used in designing the GuSu project will be the internet which will provide an endless amount of information at the team's disposal. The GuSu project team first did countless research searching the internet for ideas that that would suite the team members needs. After brainstorming and looking at projects from past UCF Senior Design groups the team decided on the GuSu prototype that is explained in detail in this document. Other projects that are comparable to the GuSu project can be found in section 4.4 of this document. Once the idea was agreed upon research using the internet began. This investigation commenced by searching for parts and devices that would fit the brainstorming ideas. The team would first search for each module that would be used in the prototype and compare them to other products of similar likeness. This method was done for each of the modules that will be used in the GuSu design. After completing this research the most favorable parts would be chosen to be used in this design. Once these parts were obtained they can now be put into the design and implementation phase of the project which will be discussed in sections 1.4 and 1.5 of the document.

1.4 Design Methodology

This section will describe the techniques of how the GuSu prototype will be designed. When designing a prototype there are a few dynamics that need to be addressed before the actual design begins. Some of these include the engineering behind the design, the marketing which will provide the product to the public and of course the actual production of the design. First off the research behind the design must be completed first which was discussed in section 1.4. After doing so, the plan can be thought about much more. One thing to point out is how the design's ability can conform to professional standards. A number of points that can be taken into consideration are those of

the safety of the design, the environmental friendliness, and how the designs specifications coincide with any governmental laws or stipulations. Once these different regulations are taken into account the design's architecture can be thought about.

The GuSu design will include various modules that are described throughout this technical document. These devices were ordered off the internet and will be put into practice during the implementation phase of the design. After receiving the parts the project team studied the various schematics and connections that would have to be made. The central part of the design will include a PCB (Printed Circuit Board) that will allow all of these modules to link together. The project team will understand the schematics of the devices and will have to create circuitry to allow these devices to talk to one another.

Team members Josh and Andrew attended a class given at UCF that would teach students to learn how to use the milling machine that would create these printed circuit boards. After having a better understanding of how this machine creates these circuit boards the team will be able to better recognize how to design the main board. Following the various schematics behind the devices will allow the project team to integrate these into the printed circuit board. Once the devices can communicate with one another the software design will be thought about. After doing much research and thinking about the design of the GuSu prototype another methodology can be put into practice, this is the implementation phase of the project which is discussed in the following section.

1.5 Implementation Methodology

This section describes the implementation phase of the design. The implementation will include the actual building and testing of the prototype. The purpose of going about the implementation step of designing will ultimately be the deciding factor in which the design will succeed or fail. As seen in the two previous sections the research was completed as well as the design brainstorming and production.

After these steps were completed the GuSu project team will now apply this background information into a real working prototype. The first step in achieving this methodology will be to assemble the purchased devices in a safe and secure environment. The purpose of building this in a safe environment will ensure the devices stay intact and also the builders. Various methods will be used when constructing the devices; such as soldering, wiring, and testing. When soldering and wiring up each device to one another as well as the printed circuit board the project team must be careful not to damage the circuitry on the devices. Simple scratching or high heat can ruin an electrical circuit in an instant.

Once the devices are connected to one another the testing phase of the implementation methodology will come into place. This phase will include all the directives needed to make sure the hardware is working properly. The project

team will use multimeters, oscilloscopes, and other electrical testing equipment to make sure there are no bad connections. As soon as this stage is completed and there are no technical errors the software design can begin. Software design will incorporate many different aspects to controlling all of the devices to perform to the team's specifications. This stage of implementation will include a lot of hard work and time to make sure the GuSu design carries out the actions described in the document.

1.6 Project Management

This section describes the aspects behind what project management means and what it can do for the designing team. The purpose of project management is a way to plan, organize, and manage resources at the team's disposal to accomplish a goal that has been set for the GuSu project team. The main goal of the GuSu project team is to successfully implement the design of GuSu and to present it to the fellow Senior Design peers and the UCF faculty. The success of the design will decide whether or not the project team was able to achieve the various principles that were instilled in the group members throughout their college career. Some of these principles that were taught throughout the college program were that of working with others and the ability to apply engineering practices and ethics. If the design meets the specified requirements the GuSu project members will know they have accomplished something meaningful and worthwhile.

To achieve this project management must be put into effect. Using project management will ensure that the team is on schedule and allow the team to make sure deadlines are met. The GuSu project team has set up an active live web folder using Windows Live Mesh which will allow the members to add and update documents, images, and information. This will allow each team member to view each other's work and use constructive criticism to make sure each member is on the right track. Google Documents are also used when updating documents. The GuSu project team is currently using the Google Docs to update information about billing and specifications. Using these methods is a way to achieve the organization and management elements of project management. To achieve the planning aspect of project management a milestone chart was set up at the beginning of the design. A milestone chart is a way of depicting key events along a period of time. Most of these events will be deadlines that the GuSu project team must meet in order to successfully implement this design. Figure 1.6.1 shows the GuSu milestone chart. Taking into account all of these factors it is key to implement project management in a plan this complex, otherwise it could be extremely difficult to meet all of the goals that have been set beforehand.

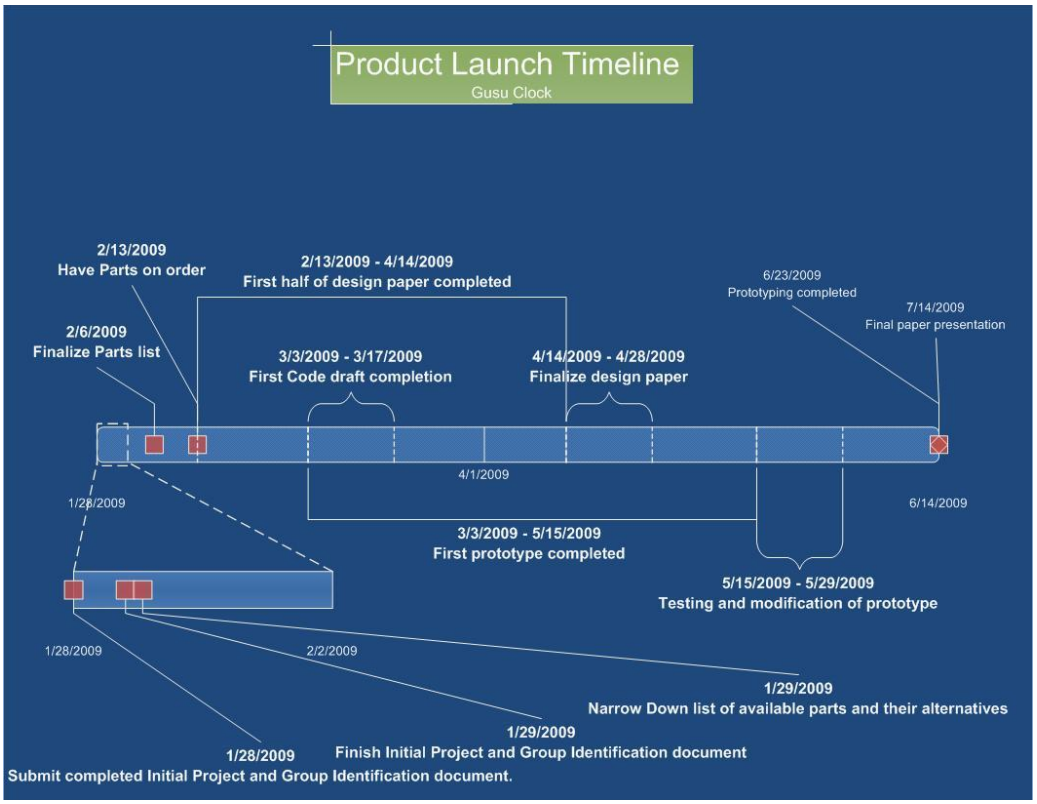


Figure 1.6.1: Milestone chart showing project schedule and deadlines.

Back in the beginning of January 2009 when the Senior Design I semester started the students were told to break into groups of three or four. All four of the group members have previously been in classes with one another and knew that this team would work well together. Every Tuesday and Thursday after the morning class the GuSu team would meet with one another to discuss project ideas and research methods. After deciding on the GuSu design the team continued to meet each week further discussing components that would be used and how the overall design would function.

The team decided that to break up the 120 page technical design document would be the best way to tackle this challenge. The document write up was evenly divided among the four group members, each choosing a module to research and design. Seen in the following list is a sample of what each member has worked on.

- Andrew: All sections related to Audio output, FM Tuner, Power and Battery backup, Printed Circuit Boards.
- Josh: All sections related to Alarm Clock External Casing, ZigBee, Coffee Machine, Microcontroller, and Clock Module.
- Matt: All sections related to LCD display, MP3 decoding, SD Card reader, Budgets, Executive and Final Design Summaries.
- Philip: All sections related to sensors, User Interface, MP3 decoding, Software design.

After deciding which team member would tackle each section all of the members worked on the design document on their spare time at home and school with meetings having continuity from week to week. The team set deadlines for how many pages would be completed with the average completion time being 5 pages per week. Keeping with this schedule the team eventually found that they were ahead of the agenda and would have time to spare at the end of the semester to deal with formatting, appendices, and company emails. The team will meet a few times before the deadline to work on formatting and proof reading of the design document. Lastly, the finished documentation will need to be printed and bound, which will most likely be done at Kinko's or the on campus printing center.

1.7 Similar Projects

There were a few similar projects that were found on the web and in past UCF Senior Design classes that correspond to the project this design will implement. The team examined all of these projects to get an idea of how to execute the GuSu system and make it more efficient.

The first project looked into was Lazy@Home [2] designed by a former UCF Senior Design engineering group. The purpose of this project was a hands free approach to controlling different aspects of electrical devices within your home. This group created a base station similar to the GuSu base station that would control features of the home such as the air conditioning/heating thermostat, the lighting, vents, and even the wall outlets. The main idea behind this project was that it allowed the user of the home to control all of these utilities in their home via web interface so if you left on a vacation you could still control all of your electrical appliances from the internet. The idea behind this was if the user forgot to turn off certain lights after you had left the user could easily turn them on or off from a remote location. Another idea behind this was that when the user was planning their return home they could easily adjust the thermostat so the home would be perfectly cooled or heated to their liking. One more interesting feature that was implemented into the Lazy@Home station was turning on and off lighting outside of the house. The GuSu project team thought this was very intriguing because it allowed their users to turn on outside lights at night so it could seem like someone was at home even though no one was there. Seen in figure 1.7.1 are two diagrams taken from Lazy@Home. The image on the left shows the Use Case diagram of the system and its respective feature components. The image on the right shows the web interface that this team created to control the Lazy@Home base station via the internet.

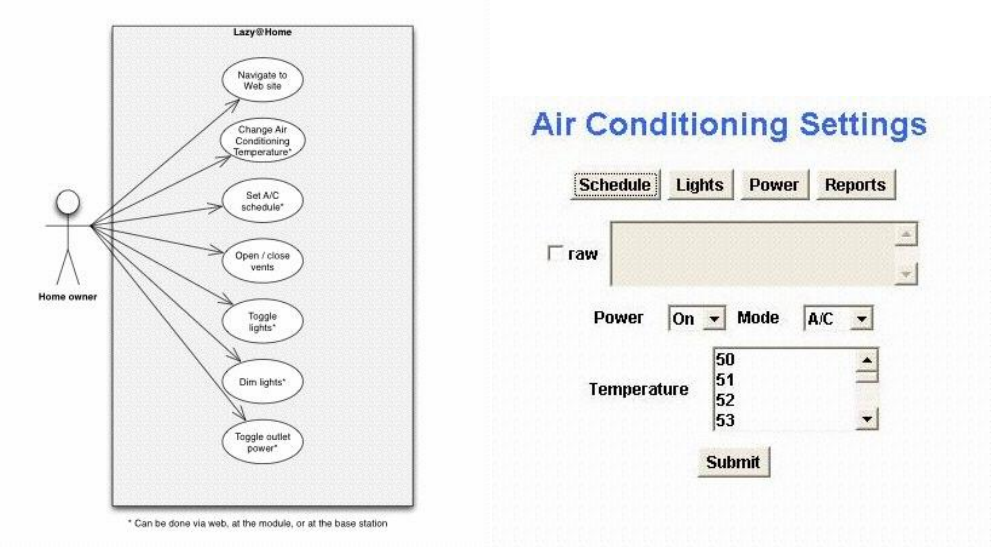


Figure 1.7.1: UML diagram and web interface of the Lazy@Home Senior Design project. Reprinted pending permission from the Lazy@Home UCF Senior Design group.

Other projects which were looked at were similar. One was from a UCF senior design and another from Columbia University. The name of the project at Columbia University was the OH800 [3], an alarm clock that consisted of a base and a ball that would shoot out when the alarm was activated. This project included a solenoid that would push the ball off of the base of the alarm clock and onto the floor which would disconnect the two magnetic switches. This would launch a DC motor to turn on inside of the plastic ball and also activate a sound alarm on the base station. An anti-symmetric disk would rotate within the ball to move it around the floor. In order to deactivate the alarm the person sleeping would have to get out of bed and pick up the ball then return it to the base station which would reconnect the magnetic switches and thus turning off the alarm. While the ball would sit on the alarm clock base station the battery that would run the DC motor inside the ball could charge via the AC wall outlet. This was similar to the GuSu design in that it required the user's action to do something to deactivate the alarm.

The last project that was looked into that was especially similar to the GuSu design was the PerfectSleepSystem [4] which was also designed by a group in the UCF engineering program. The purpose of this system was to utilize electronic sensors attached to a person's body to optimize their sleep occasions. The PerfectSleepSystem contained sensors that would monitor the heart rate, temperature of the body, and also their movement. The goal of this system was to use those features to calculate which settings were most favorable for a good night's sleep. The PerfectSleepSystem was similar to Lazy@Home in that it would control external electronic devices to allow the user to sleep more sound. Examples of this would be turning off appliances such as the TV or lights. One feature of the PerfectSleepSystem that was found most interesting was the ability to detect the amount of light in the room. Having this feature would allow the

base station to turn on its alarm clock to wake the person up at a certain time; whether it is in the morning, afternoon, or at night. The PerfectSleepSystem was fascinating as it could detect when to wake the user up when they entered the light sleep mode based on their movements and heartbeats. An example of the base station is shown in the left image of figure 1.7.2 and examples of where the sensors were placed are shown in the right image.

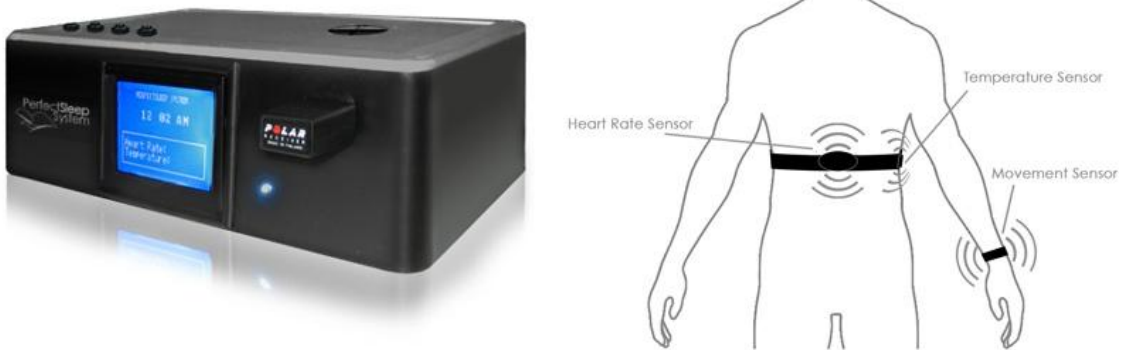


Figure 1.7.2: Base station of the PerfectSleepSystem and a diagram of the sensors involved. Reprinted pending permission from the PerfectSleepSystem UCF Senior Design group.

After reviewing both of these projects and many others it was found there were similarities in their design compared to the GuSu design. The Lazy@Home system would use the Zigbee protocol as well as the GuSu design and all of these systems would utilize the concept of wireless technology to achieve a common goal. Another resemblance between these systems is the base station. These designs would use the concept of a base station as the central control unit including the design of GuSu. The purpose of this base station would allow all features of the designs to be easily integrated together. The main difference that can be seen from these designs compared to the GuSu design is the fundamental function of what each system performs, each system having a different goal to accomplish. Once these designs were looked into the GuSu project team got a better feel for how this design would be implemented and put into effect.

2.0 Case Design

The alarm clock needs to be enclosed in an aesthetically pleasing and durable case. It will have a single SD card slot and removable back panel for access to the battery compartment. The three options investigated are an ABS plastic box, Plexiglas (Lucite) sheets, or a wood case.

2.1 ABS Plastic Enclosures

ABS plastic boxes are readily available from online retailers and local home hobby stores such as radio shack. They are sturdy, water resistant, and durable. They are also non conductive. The cases are available in clear to opaque colored boxes and available in a variety of sizes (figure 2.1.1).

Figure 2.1.1: Sample ABS enclosures from Polycase Inc.
Reprinted pending permission from Polycase Inc.

Pre built boxes are designed with holes for switches and buttons while providing thin walls allowing easier mounting of said buttons and switches. This is also a drawback for ABS boxes. The enclosure size of the box is also static and the PCB layout has not been finalized at this point. The design calls for a device size of 12x4 but this could change based on further PCB designs and additions to system specifications. The design calls for buttons to be mounted in a circular fashion which in our research is not available. Many online retailers offer custom ABS enclosures but the costs are prohibitively expensive.

2.2 Lucite Sheets

With Plexiglas sheets, the box can be constructed to our specifications. It can be modified easily and adjusted to the final design of the system. A solvent or adhesive is applied to a finely sanded edge and the solvent is allowed to cure for 12 hours. The edges are then sanded to make a smooth edge. Another option is using an extruded acrylic adhesive like Weld-on #5. It is applied as glue and the seams are taped allowing it to dry for 12 hours. While simple to implement, the aesthetics are certainly compromised with this type of implementation. The available Lucite products are clear and the user would be able to see the

internals of the system. While some would prefer the look of the clear case, the group feels this would limit its appeal to a select few.

2.3 Wood

Wood sheets are readily available at local home improvement stores in a wide variety of species. An oak or a maple cabinet grade sheeting with different stains would certainly provide an aesthetic appeal and are easy materials to use in construction. Small half inch sections are available for cabinet construction and can be formed and resized to our specifications. Simple wood glue and finishing nails can be used to construct the clock. Holes for buttons can be milled to any specification and mounting hardware such as standoffs for the PCB can be drilled directly into the case. The only drawback to the wood design is that it is flammable. In order to limit the risk of fire, ventilation shafts and plenty of open air space can be provided within the enclosure. Adequate heat sinks on the required components and if necessary cooling fans could also be installed although this should not be required. The alarm clock system also has a battery backup in which the battery will be internal to the clock. The access door will be a simple plate attached with two small screws. A sample design is provided in figure 2.1.3.1.

The design calls for external interfaces for the SD card and external plugs for external sensors. The plugs will be modular allowing digital interfaces and analog interfaces. The plugs will have different adaptors for either the analog or digital sensors to prevent the end user from inserting the wrong type of adaptor. A single female two pole receptacle will pass through the case to power all internal devices. A grill will be created on one of the sides for the speaker output. If stereo speakers are implemented in further designs, two grill plates will be created on both sides of the OLED screen. The symmetry will maintain the aesthetic look of the system.

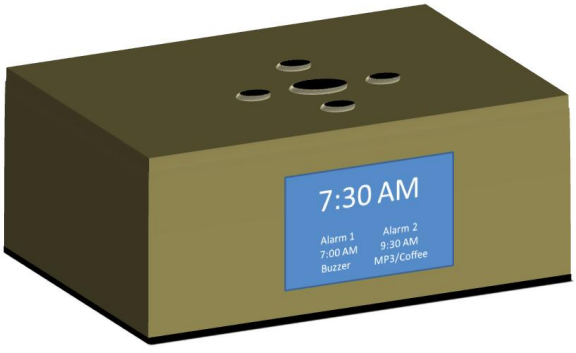


Figure 2.1.3.1: Sample Clock Case Design

There will be five buttons on the top and a hole for the OLED screen in the front. The LCD screen has mounting holes built directly into the PCB and will be screwed directly to the front face plate. The buttons may require straps through the terminals to hold them in place. Another option would be to make the top plate a little thicker ad counter sink holes to hold the buttons and pass the wires through. In order to make the buttons visually appealing, wood buttons will be overlaid and glued to the off the shelf plastic buttons.

The cost of a wood enclosure is second to ABS in expense for material in the alarm clock design. Suitable plastic ABS enclosures range in price from $6.50 to ~ $20.00. Wood and Lucite are difficult to estimate based on possible errors that would require extra materials. Assuming no errors in construction, a single piece of ½ inch 4'x4'' cabinet grade oak or maple would be sufficient. The cost for this piece of wood is around $10.00. The glue and nails would add on an extra $5.00.

3.0 Microcontroller

3.1 Technical Objectives and Specifications

3.1.1 Goals

The microcontroller is the core component of any embedded system and it is vital that this device be chosen with care. It must meet all the system requirements and as well be affordable in regards to the physical device and the development costs.

The main microcontroller will interface all of the devices within the alarm clock (figure 3.1.1.1). This includes any sensors such as passive infrared sensors (PIR) and/or pressure sensors, the real time clock, the MP3 decoder, the XBee module, and the . The microcontroller must accomplish the extensive interfacing and be capable of quick responses for the event driven functions.

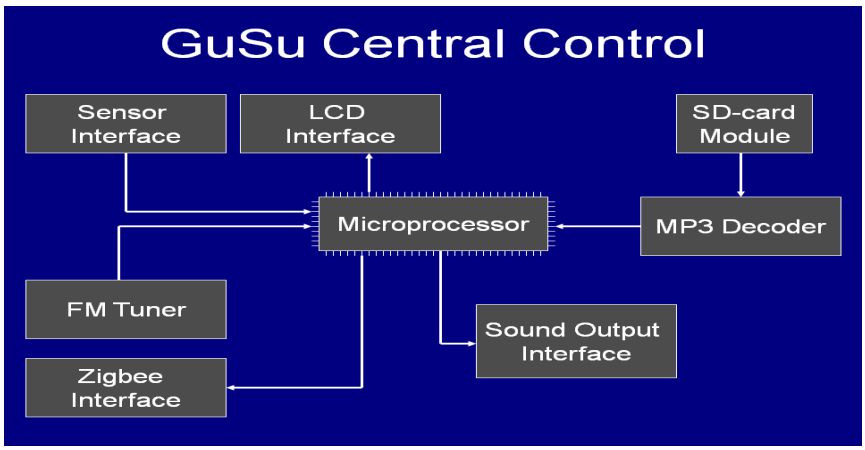


Figure 3.1.1.1: System Block Diagram

3.1.1.1 Input/output

For the PIR sensor module, the microcontroller must have a single digital input pin. Since the PIR sensor module is digital, only a single digital input will be required to monitor the status. The pressure sensors also provide digital outputs and will require two pins for each sensor. The graphical LCD the group has chosen uses UART as does the XBee module so these two devices will need two separate UART pin sets. The MP3 Player and the Real time clock interfaces are over SPI which is a shared port so the system will need only one set. There will also be a minimum of six buttons for user interface and each of these will use a single digital input.

3.1.1.2 Power

To simplify circuit design, devices were sought out which will operate in similar voltage ranges. The common range between devices is 3-5 volts but some components require more so a voltage regulator will be necessary.

3.1.1.3 Software

Since there will be devices communicating wirelessly to separate microcontrollers the system architectures must be similar. The ability to use the same code for both microcontrollers will reduce the amount of code, which will lower the time it takes to develop the software, as well as debugging and maintenance. Specifically, the selected microcontroller must have an Integrated Development Environment (IDE) available for it which allows for programming in the C or C++ programming languages. In addition to a common programming language, the control panel and the microcontroller must use the same bit architecture.

3.1.1.4 Physical Package

Both of the microcontrollers will be installed on single sided printed circuit boards. Each device will be hand soldered to the boards. In order to make it easier to mount and solder the microcontroller, they must be available as a dual in-line package (DIP).

3.2 Research

After taking into consideration all of the requirements, the range of choices for the microcontroller were narrowed down to two types. The first choice was from the Texas Instruments MSP430 microcontroller family. The second choice was the Atmel AVR line of microcontrollers. The choices were limited to these devices because of the access to both of the development kits for these, as well as some experience in developing code for them.

3.2.1 Texas Instruments MSP430

The first and foremost problem with the MSP430 is the availability in packaging. Texas Instruments provides a wide variety of chips in MSP430 family. The only problem is that they are all only available in a TSSOP and QFN packaging. The preferred package type is DIP. The group chose to explore the option of using the MSP430 with QFN adaptors or sending off PCB and microcontroller for soldering. The specific device investigated for the project was the MSP430F2274IRHA. The reason for choosing this device is that it met all of the requirements excluding dual UART ports and Texas Instruments currently sells a development kit which is integrated with a ZigBee module using this chip. The device features a 16-bit RISC CPU, 16-bit registers, and constant generators for code efficiency. Wake up time from the 5 low power modes is less than 1 µs. It has two built-in 16 bit timers, a single UART module, 10-bit A/D converter, and two integrated operational amplifiers. It also supports SPI and I^2C interfaces. There is an internal low power oscillator which operates at 32 kHz and built in support for an external RTC. There are 32 I/O pins available. In order to use this microcontroller the group would have to investigate the use of another LCD. With 32 I/O pins available, the LCD could easily be supported using TTL logic instead of the UART interface. Although, controlling a graphical LCD with TTL logic may add another degree of complexity to the project. The MSP430 operates at 1.8 – 3.6V which is the same voltage as the XBee radios.

3.2.2 Atmel AVR

Atmel manufactures 8 bit microcontrollers called ATmega. This microcontroller is a high performance, low power consumption, 8 bit RISC microcontroller [5]. This chip is an ideal candidate for embedded applications since it is well supported by both its manufacturer and the market. They offer multiple devices with program memory in the range of 4 - 256 Kb, data memory from 256 – 2048 Kb, and packages which range from 28 - 100 pins. The ATmega can operate at as high as 20 MIPS at a clock rate of 20 MHz although the system will be using a real time clock with timers; the chip also has the counters for use in embedding timing functions. Both internal and external interrupts are supported which will allow event driven communication over ZigBee and sensor monitoring. The required voltage supply can be anywhere from 1.8 to 5.5 volts with a current drawing between 250µA to .1µA which falls in the range of other modules within the system. This will lower costs by lowering costs in power system design. ATmega also support boot loaders which will permit remote upgrading and there are also multiple resources available online for working with Atmel chips and they are inexpensive. Development Software is also provided free of charge.

3.2.2.1 Input/output

There are multiple members of the Atmel mega family of microcontrollers that have at least 20 input/output pins, two serial interfaces and an SPI interface. The maximum frequencies supported by these chips ranges from 4MHz to 20MHz.

3.2.2.2 Power

As illustrated in Figure 3.2.2.2.1, the operating voltage for all of the chips is either 1.8V-5.5V or 2.7V-5.5V. The difference in minimum operating voltages varies depending on the chip selected. The Atmel AVR microcontrollers include a 32 kHz crystal oscillator for use during Power Save mode. This lowers the current consumption during Power Save mode down to 650nA. Since the coffee machine will not be running continuously, the microcontroller will be able to be placed in power save mode and have wake up events based on button presses or randomly to check for new information from the alarm clock

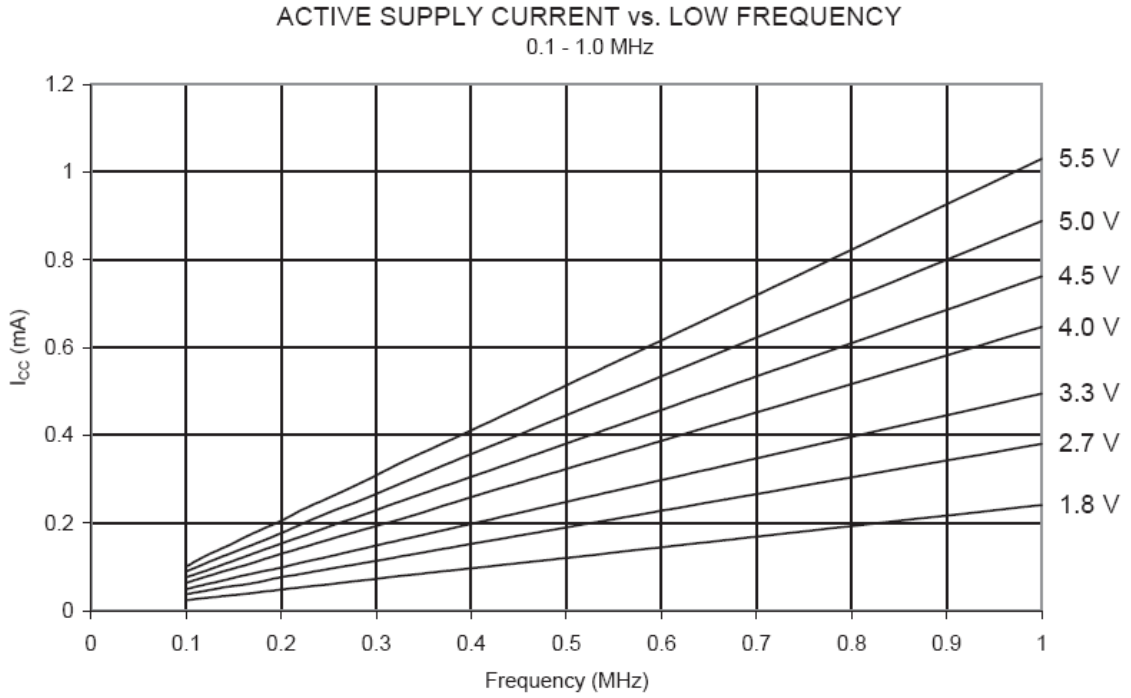


Figure 3.2.2.2.1: Supply Current Requirements
Reprinted with permission Atmel [5].

3.2.2.3 Software

Atmel provides a free IDE that can be used for writing and debugging applications written for all of its chips. The IDE is also used to compile and flash the microcontroller. It also includes a simulator that can be used to aid in application development and testing. It allows for program writing in C, Pascal, BASIC, and assembly. There is also an open source IDE called Arduino/Sanguino [6] [7]. It enables descriptive coding and simplifies the programming. The boot loader is compatible with both the ATmega164 and the ATmega644P. Atmel's IDE and Arduino's both have extensive support available online.

3.2.2.4 Physical Package

Both megaAVRs are available as dual inline packages. This makes these chips good for use in the project because soldering these chips can be done by hand, rather than being sent out. This will keep the cost of using these particular chips low and ease prototyping.

3.2.2.5 Summary

All of the Atmel megaAVR microcontrollers meet the specified minimum project requirements. The availability of each chip as a DIP is advantageous, which will keep the cost of using them low. Since there are multiple free choices for IDEs, software development costs are also very low.

3.3 Implementation

Based on the information in the research presented in section 3.2, an Atmel microcontroller will be selected for use in this project. Atmel site lists three ATmega AVRs which meet our requirements for the clock and nearly all of the chips are compatible with the specifications for the coffee machine. For the clock, the ATmega644p will be used due to the fact that is widely supported by Atmel and by the open source Arduino boot loader. Because the software that will be used on the microcontroller has not been written yet, the amount of flash memory needed on the microcontroller is unknown. The ATmega168 has 512 bytes of EEPROM memory available, and if this proves to not be sufficient, then the ATmega644p will be used in place of it. The use of a different chip will require virtually no additional effort because the software written for Atmel microcontrollers can be recompiled to be used in any other Atmel microcontroller. If the size of the software for the microcontroller exceeds the maximum flash memory size of 256Kbytes available in the megaAVR line, then expanding the flash memory using an external chip will be researched. The pin configuration of the ATmega644p (figure 3.3.1) is shown below.

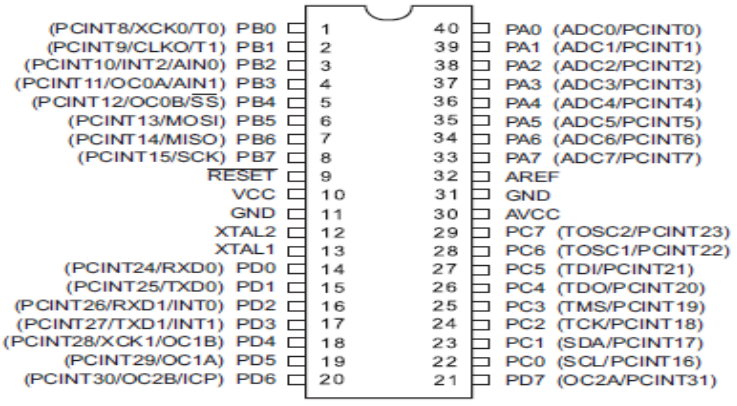


Figure 3.3.1: Pin configuration for ATmega644p.
Reprinted pending permission from Atmel.

The microcontroller needs the ability to manage all of the devices within the alarm clock system. It also handles the control of the coffee machine over the XBee radio modem and the detection of the remote sensor unit. Table 3.3.1 outlines all the required pin assignments for the interconnections between all secondary control modules. The location of the pins on the ATmega is in figure 3.3.1.

Device	ATmega644P	Device Pin
MUX	Pin 1 (D0)	Pin 5 (MP3 Control)
MUX	Pin 2 (D1)	Pin 2 (FM OUT)
Buzzer	Pin 3 (D2)	Power Buzzer (BZCTRL)
Clock - DS1305	Pin 4 (D3)	Pin 10 (CE/CLKSS)
Clock - DS1305	Pin 6 (MOSI)	Pin 13 (SDO/CLKXMIT)
Clock - DS1305	Pin 7 (MISO)	Pin 12 (SDI/CLKRX)
Clock - DS1305	Pin 8 (SCK)	Pin 11 (SCLK/CLKSRL)
Clock - DS1305	Pin 12 (XTAL2)	Pin 15 (32 KHz/Clockout)
MP3 (Data) - STA013	Pin 5 (D4)	Pin 7 (BIT_EN)
MP3 (Data) - STA013	Pin 7 (MISO)	Pin 5 (SDI)
MP3 (Data) - STA013	Pin 8 (SCK)	Pin 6 (SCKR)
MP3 (Control)		Pin 3 (SDA)
MP3 (Control)		Pin 4 (SCL)
LCD	Pin 14 (RX0)	Pin 5 (LCDRX)
LCD	Pin 15 (TX0)	Pin 3 (LCDXMIT)
XBee	Pin 16 (RX1)	Pin 2 (DOUT)
XBee	Pin 17 (TX1)	Pin 3 (DIN)
Button - up	Pin 35 (D26)	Button
Button - Down	Pin 36 (D27)	Button
Button - Left	Pin 37 (D28)	Button
Button - Right	Pin 38 (D29)	Button
Button - Select	Pin 39 (D30)	Button
PIR	Pin 40 (D31)	Digital Out

Table 3.3.1: Pin Assignments for ATmega644P

3.4 Test Plan

This will include testing of the sleep function, sending data serially, and receiving data serially. The sleep function will be tested by programming the microcontroller with a simple application that will put it to sleep and wake it up after a given amount of time. During this test, an amp meter will be used to monitor the amount of current being used by the microcontroller. If the results from the amp meter reflect the appropriate values expected for sleep mode and active mode operation, then it passes the test. However, if the current draw of the microcontroller does not indicate that it has gone into and woken up from sleep mode, then it fails the test. The sending of serial data will be tested by using the

development board serially linked to a computer (main panel). A simple application will be written and programmed into the microcontroller that will cause it to send predetermined data serially out of one of its ports. A terminal on the computer will monitor the port and output what it receives over that line. If the data received by the computer matches the data sent by the microcontroller, then the test passes, otherwise the test fails. The receiving of serial data will be tested by using the development board serially linked to a computer. The computer will send data serially, and a simple program on the microcontroller will receive the data over the specified port and store it in RAM. Monitoring software will then be used to read the value stored in RAM. If the value matches the data sent by the computer, then the test passes, otherwise the test fails.

4.0 Alarm Module

4.1 Block Diagram

The alarm module will consist of an FM tuner, an MP3 decoder, an SD Card Reader, an audio amplifier, a speaker, and a buzzer. A multiplexer will need to be used to determine which audio output passes to the audio amplifier, and it will be controlled by the microcontroller, as will the buzzer. A block diagram of the Alarm Module is shown in Figure 4.1.1.

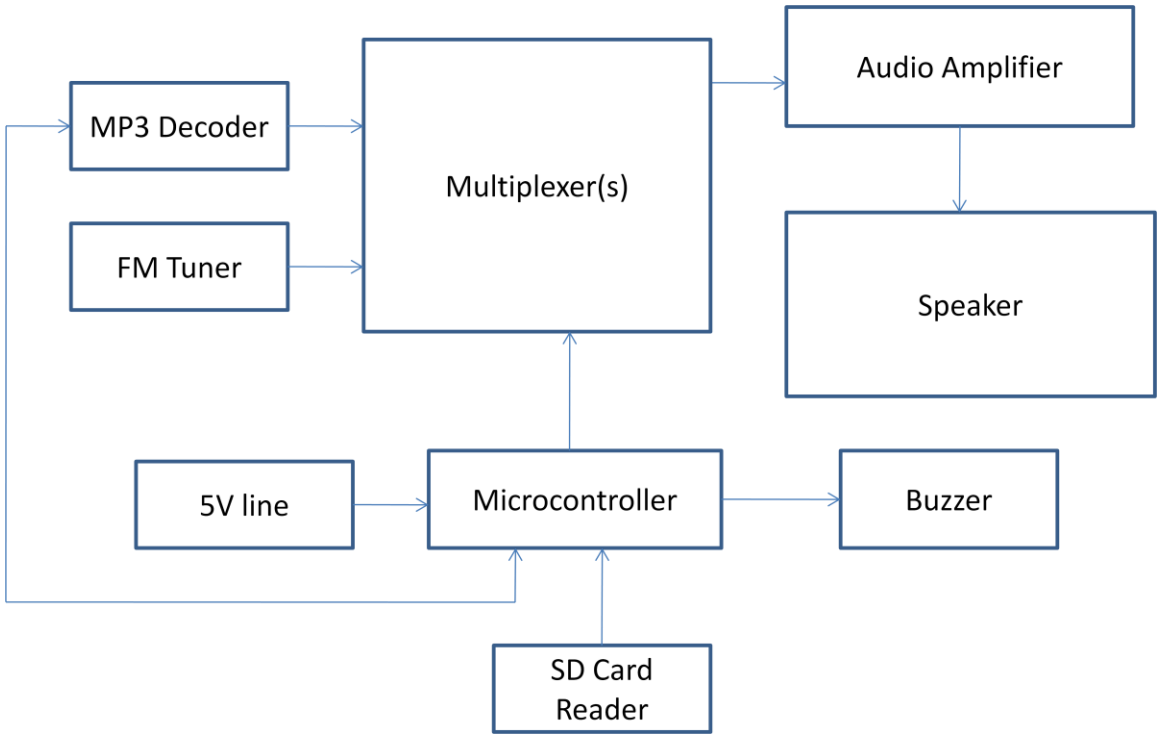


Figure 4.1.1: Block Diagram of the Alarm Module.

4.2. Design Requirements

4.2.1 Audio Output

There will be a standard buzzer alarm for the clock that can be used. The microcontroller will be in charge of selecting which output (mp3, standard buzzer alarm, or the FM radio) will be accessed. If it is the buzzer, it will buzz. The other two would go through an audio amplifier. This amplified output would then be pushed through to the speaker.

4.2.1.1 Standard Buzzer

Some small amount of research was needed to be done for the common alarm buzzers. There are several different types, but most of the common ones use a DC voltage input and output a buzz. Each of them offer different levels of sound output, measured in decibels (dB). 90 dB should be safe for use of under 8 hrs, and 85 dB and under would be safe for much longer (it is an exponential relationship) [8]. As such, the alarm buzzer could easily exceed 85 db without causing hearing loss as it would only be on for a short time.

A quick search on Digi-key.com showed many different buzzers. A 5mm 2.8 kHz PCB buzzer with sound output of 85dB minimum was found (part number CEM-14R06CT) for $1.24 each. It runs on 40mA. A problem with this model is that it requires 1.0V to 1.7V DC, which is not in the current plans for the clock. Of course, a step-down could be used to achieve this voltage, but if it were not necessary then it would be simpler to avoid that. Further searching found a 12mm (slightly larger) 2.3 kHz PCB buzzer with a sound output of 85 dB minimum and a maximum current of 30mA. This was the WST-1205S, manufactured by Soberton Inc. It requires a supply voltage of 4-6V, which would be ideal for this group. It runs $1.81 each at Digi-Key.com.

The buzzer will be activated whenever a voltage is passed through it. Because of this, the microcontroller must pass the 5V whenever the buzzer is needed. A schematic of the buzzer is shown in Figure 4.2.1.2.1, and the 5VM is controlled by the microcontroller.

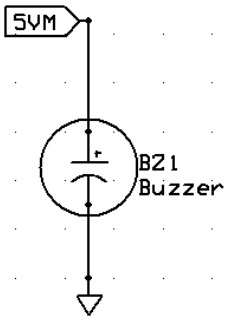


Figure 4.2.1.2.1: Buzzer schematic.
Schematic was made using ExpressSCH.

4.2.1.2 Audio Amplifier

Once the input is chosen, the audio will need to be amplified. This could be done by a simple non-inverting setup of an Op-Amp (operational amplifier). The most common Op-Amps used are the 4558 or 1458 models, which are available for roughly $0.29 to $0.49 each. There are many different brands of these, but they are all of the same utility. The gain would of course be dependent on the desired audio output, and on the audio inputs. The gain cannot be large enough to saturate the Op-Amp, though, which would occur around 10 or 11 V if the Op-Amp was biased at 12V. Figure 4.2.1.2.1 below shows a basic non-inverting Op-Amp biased at +-12V, with a gain of 100. This gain was found using basic electronic formulas learned in EEL 4314, Electronics 2 The equation for it is Vout = Vin * (1 + (Rf/Rin)) = 10Vin. Gain is defined as Vout/Vin, so would be equal to 10.

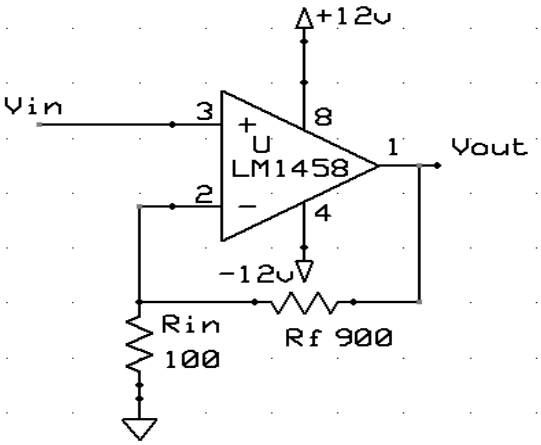


Figure 4.2.1.2.1: A non-inverting Op-Amp with a gain of 10. This schematic was created with ExpressSCH.

A problem exists, however, with using a + and – power supply. As of now, the group is planning on using a +12V DC power supply that will be converted to the various necessary voltages, i.e. 5V, 3V, etc. However, there are no plans to have a -12V line. There do exist Op-Amps that can take a positive DC voltage and ground, with no negative DC voltages required [8]. However, these Op-Amps can only amplify an AC signal with only positive values (any negative signals are eliminated). For obvious reasons, this will not work as an audio amplifier, as audio signals use both the positive and the negative sides of the signal.

There are workarounds to this obstacle. Using one battery, it is possible to 'trick' the Op-Amp into being biased as both a positive and negative bias. This can be done by using a voltage divider, with equal resistor values, and grounding the zone between the two resistors [8]. For example, if a 12V battery were used, there would be considered a +6V and -6V on either side of the battery, biasing either side of the Op-Amp, if the circuit in Figure 4.2.1.2.2 were used. The

problems with this approach, however, are twofold. For one, the range of the Op-Amp will be limited by the + and – 6V. This shouldn't be a real problem in this group's example, however. This bigger problem is that the load of the Op-Amp will be in parallel with the resistors. So if the negative or positive biasing of the Op-Amp draws more or less current (which will be the case for most of the time of operation), it will offset the loads, making them not exactly +6V and -6V. Depending on the operations, it could be +5.7V and -6.3V at times, and other variations at other times. This likely would also not have too large of an effect on this project, although in general it is not a heavily recommended option to use for biasing.

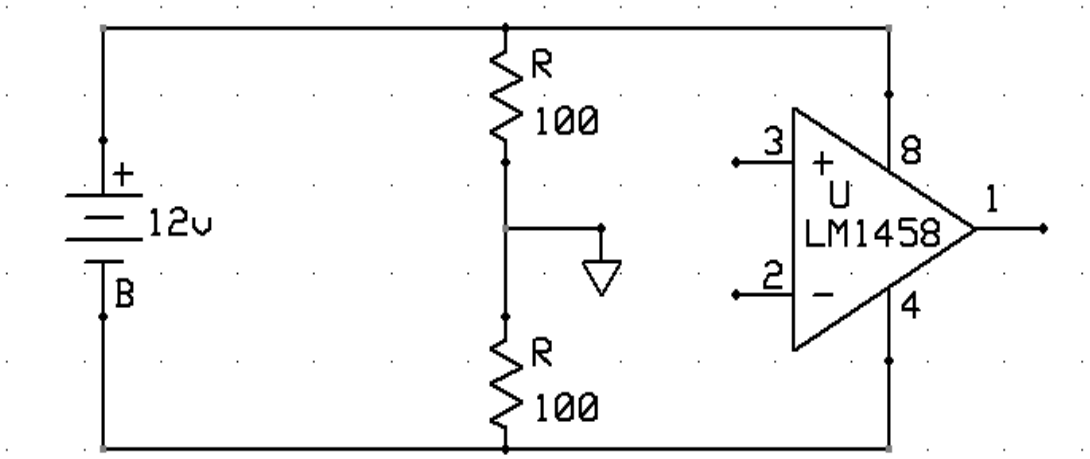


Figure 4.2.1.2.2: 'Trick' for biasing the LM1458 Op-Amp. The top rail voltage would be +6V in this example, and the bottom rail would be -6V. Schematic created with ExpressSCH.

Because there will already be a 12V power rail, however, the simplest method (if another battery would have to be used anyways, as in the above 'trick' method) would be to simply use another 12V battery. This battery would just be upside down (positive terminal to ground, negative terminal to the negative biasing of the Op-Amp). This is a good method in that the range will be from roughly 11V to -11V, which should be more than ample in this project. The downside, is that it would require another battery that would need to be replaced over time, in order to keep the Amplification, and thus the point of the clock, running. It would still be simpler than attempting to plug in another 12V wall wart into the outlet, and running that voltage to the clock as a negative line (which would be another option). The final schematic for the audio amplifier to the speaker, that will be used in this project (using the LM1458 Op-Amp), is shown in Figure 4.2.1.2.3, along with the necessary second battery for the -12V. Refer to section 9 which details the GuSu alarm'sto the +12V line being used. The audio input is coming from the FM Tuner, Mp3 reader, and standard alarm buzzer. Though the resistor values are listed to give a gain of 10, this can very easily be changed to however high is wanted. This will be determined when all the components are in, and the

voltage and audio levels can be tested. If, for example, a gain of 100 was desired, the Rf could be replaced with a 9900 ohm resistor.

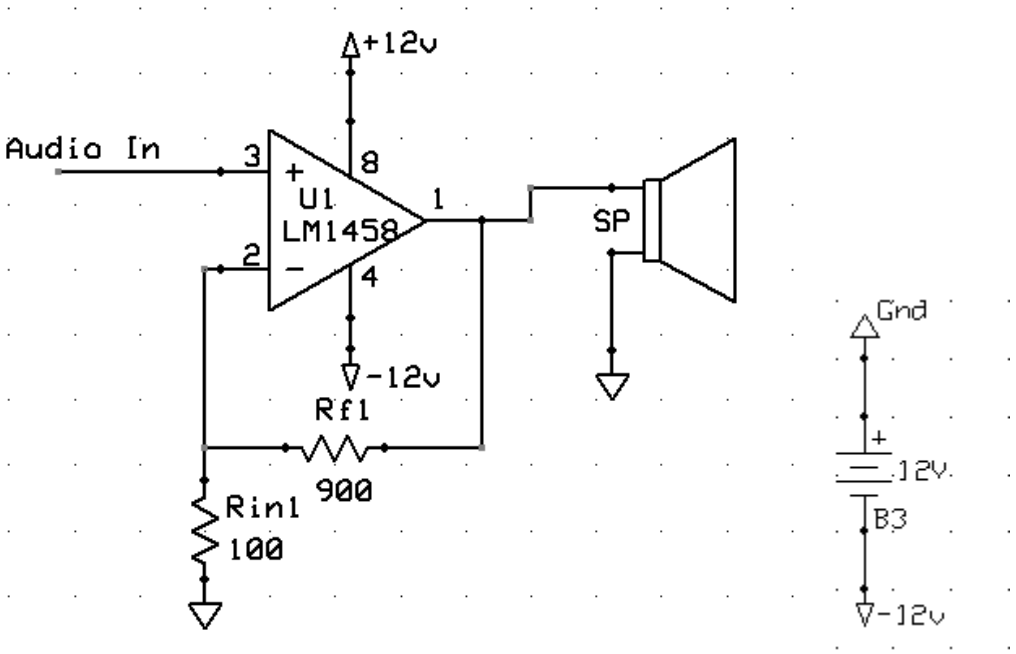


Figure 4.2.1.2.3: Audio Amplifier Module final schematic. The Audio In will be controlled by the microcontroller. Schematic created with ExpressSCH.

The audio amplifier can easily be tested. By using a constant signal (perhaps an mp3 tone), a voltmeter can be used to test the voltages at the input and at the output. The output voltage level should be roughly the input level multiplied by the gain. Of course, an aural check can be conducted by running the input through the speaker, and then running the output through the speaker to hear the difference. The PCB schematic integration will be convered in chapter 10.

4.2.1.3 Speaker

There are many types of speakers available, and through searching on Digi-Key, there were many that could fit the needs of this group. Ideally, the speaker would use a minimum of power/current consumption, output a maximum level of dB, be as small as possible, be easily integrated into the alarm clock, and have as low of a cost as possible. Of course, there are no speakers that can deliver all of those competing ideals, but a decent speaker was found by way of the Panasonic EAS-4P15SA. This speaker uses .1W max power, can output up to 92dB, has a 45mm diameter and 18mm height, terminates in solder pads, and costs $4.32 each when purchased without a bulk discount. This speaker should be more than sufficient for the needs of this project.

4.2.1.4 Multiplexers

A multiplexer (mux) will be needed to decide the audio output. Two 2x1 muxes could be used to determine the audio output, with each mux acting as a switch. Each switch would be between an audio signal or ground. These audio inputs are the mp3 decoder output and the FM tuner output. After doing some searching, an optimal 2x1 mux seems to be the TS5A23159DGSR, made by Texas Instruments. On www.mouser.com, these run only $0.81 each. Each one also comes with 2 muxes on it. A small chip in schematic form is shown in figure 4.2.1.4.1.

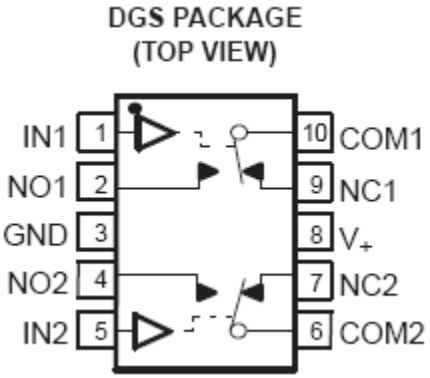


Figure 4.2.1.4.1: The TS5A23159DGSR datasheet drawing. Picture reprinted with permission from Digi-Key.

V+ would be connected to +5V (the acceptable range is 2V to 6V). IN1 and IN2 determine whether or not to pass the voltage through. When the IN1 is high, then the signal connected to NO1 will connect to COM1. When IN1 is low, COM1 will connect to NC1. Similarly, when IN2 is high, NO2 connects to COM2. When IN2 is low, COM2 connects to NC2. So, IN1 and IN2 need to be connected to the microcontroller to determine when each signal should pass through, NO1 can connect to the FM Tuner output, NO2 to the mp3 decoder output, NC1 and NC2 to the ground (so that when the devices are not activated, there will be no audio output), and COM1 and COM2 to the input of the audio amplifier. A final schematic of the mux is shown in Figure 4.2.1.4.2.

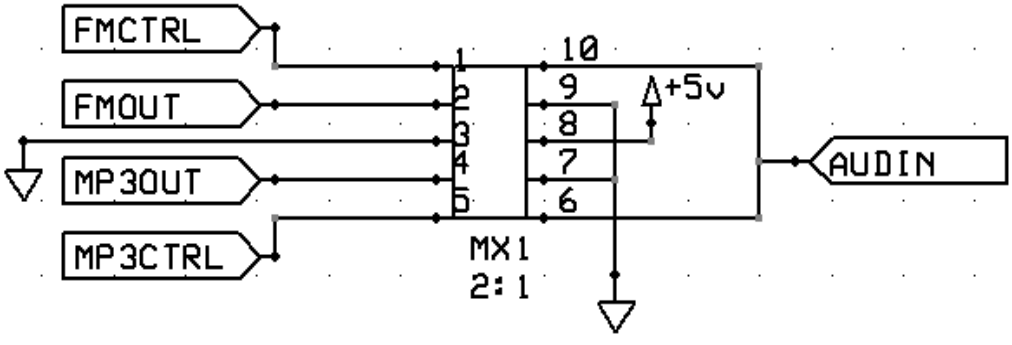


Figure 4.2.1.4.2: Final schematic of the implementation of audio amplification control. Schematic created using ExpressSCH.

4.2.2 FM Tuner

An FM tuner is to be integrated with the GuSu alarm clock and will be able to wake the user with any chosen FM radio station. This will be in addition to the option for mp3 audio from the on board SD card slot and standard alarm beeping. It is intended parts, and the radio should be able to be tuned by the user to any local station that has strong enough of reception to be easily picked up. In this case, that would be all FM stations that are broadcast from the Orlando area. The tuning needs to be relatively easy for the user to do; with a user interface on the outside of the alarm clock housing unit allowing access to change the stations. This could be done with an analog knob or two buttons, depending on the method of reception. The radio should be able to tune at every 200kHz frequency interval from 87.5 MHz to 107.9 MHz, which is the frequency range for broadcast FM radio in America. It is also important that the radio is properly selective. The radio will also output to the LCD screen the radio station that is being played when it is on.

4.2.2.1 Research

The process of integrating an FM Tuner into the clock begins with identifying the various alternatives that are available. Through researching online, it became apparent that there were two main ways to do this. Several circuits could be built from scratch and connected, or a partially or completely prebuilt receiver could be purchased.

There are many different circuits that could be built from scratch, from a (somewhat) simple single transistor receiver to more complex super heterodyne receiver circuits. The most common radio receiver today (and it has been for well over 50 years) is the super heterodyne receiver (short for supersonic heterodyne receiver) [9]. A block diagram of the important elements of this receiver is shown below in figure 4.2.2.1.1.

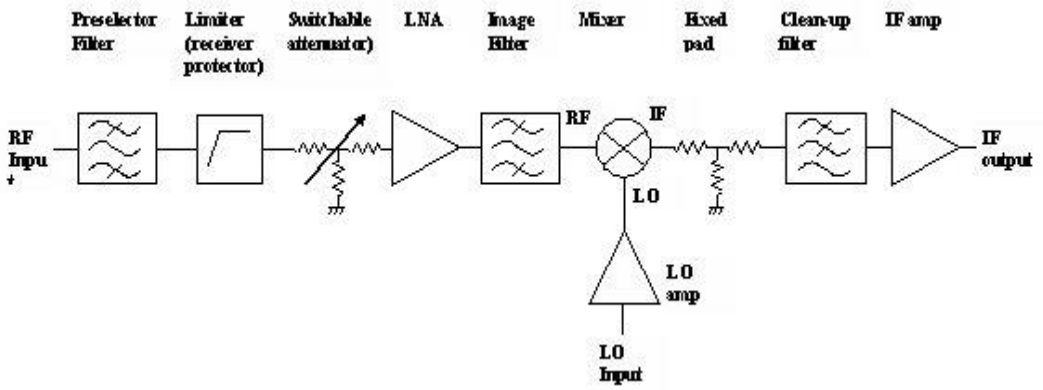


Figure 4.2.2.1.1: Block diagram of a common super heterodyne receiver. Reprinted with permission from Microwaves101.

The RF input, received from an antenna, is fed through a filter (the Preselector Filter) to severely limit the frequencies passing through to a narrow band including the frequency that is wanted. The Limiter acts similarly to a surge protector; if a seriously high powered signal were to come through the RF input, this limiter would protect the LNA (Low Noise Amplifier). The Switchable Attenuator is able to, as needed, slightly reduce the strength of the signal if it is strong enough to saturate the LNA. Next, the LNA amplifies the signal, though it is important that it is not damaged by too much power hitting it. Any damage done is permanent. The amplifier's purpose is to amplify the main signal to a point where it is further distinguishable from the background noise). The Image Filter reduces image noise, and the Mixer converts the RF signal to an IF (Intermediate Frequency) signal.

This is the main advantage of the super heterodyne receiver. The IF signal is at a much lower frequency than the RF signal, and thus, can be modified with lower-frequency components. It is much 'cheaper' to add gain to IF signals (and lower frequency signals in general) than higher frequency signals. It is also easier to be more selective with the lower frequency signal [9]. After being mixed into an IF, the signal must pass through a LPF (Low Pass Filter), the Clean-up Filter to remove any remaining higher frequency signals, such as the LO (Local Oscillator) or RF. Finally, the IF Amp is used to amplify the signal to its final level. Again, this is much cheaper to do after modulation than at its higher-frequency state. An example of an early super heterodyne receiver schematic is shown below in Figure 4.2.2.1.2.

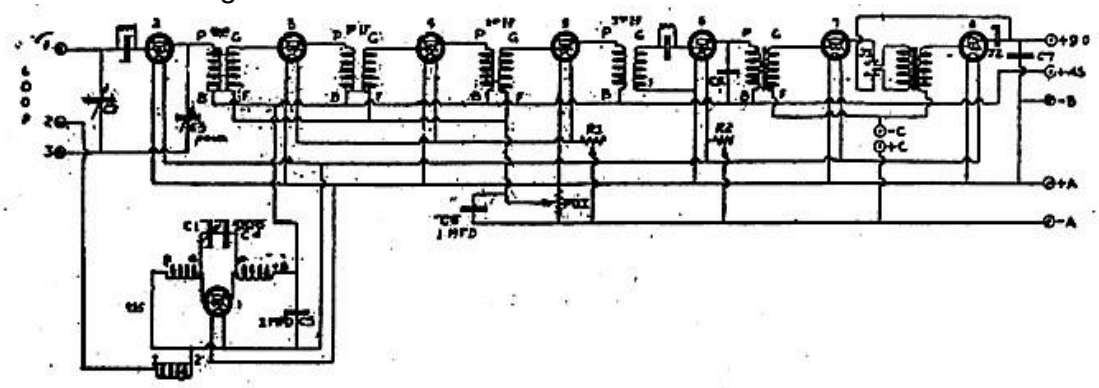


Figure 4.2.2.1.2: Schematic of an early Super heterodyne FM receiver. Reprinted pending permission from Hans Egebo.

Regenerative receivers are also an option for many who are building receivers; however, from initial research it appears that they would not be well-suited to this project. Although some of the receivers have relatively few parts (one of the design criteria is for minimal parts required; there are some one-transistor regenerative receivers), the frequency selection would not be easily implemented. It would be quite difficult to design a method for easy user control of the radio station. An example of a regenerative receiver is shown in figure 4.2.2.1.3.

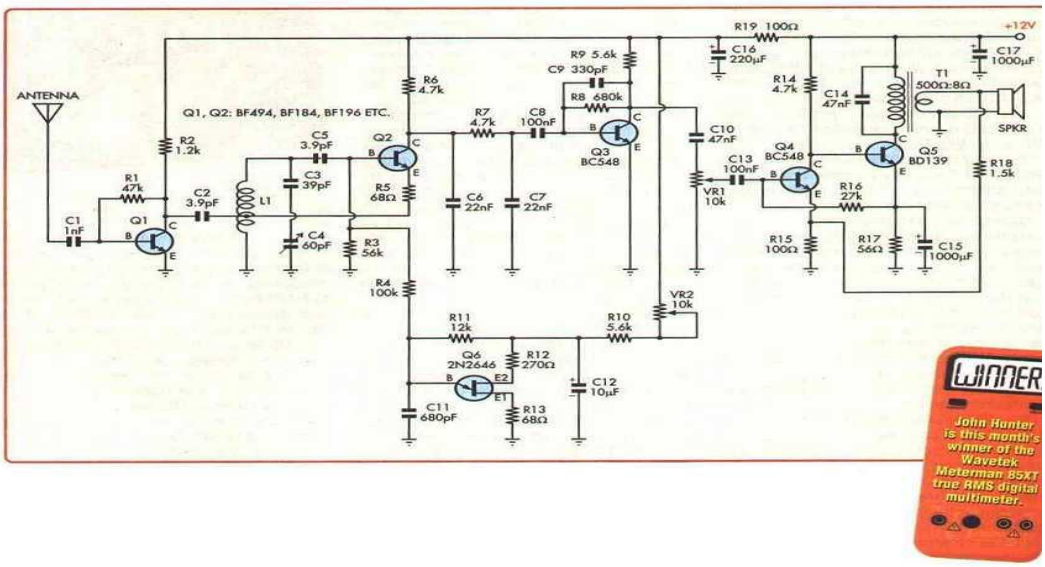


Figure 4.2.2.1.3: A sample schematic of a regenerative FM receiver. Reprinted pending permission from Electronics-DIY.com.

These circuits would not have to be fully implemented on a PCB. There are ICs (Integrated Circuits) available that consist of regenerative FM receivers. The most common one over the last 20 years may be the Phillips (now NXP Semiconductors) TDA7000. This IC is a basic FM receiver, with only mono audio output (which would be acceptable according to our one-speaker specifications), and selectivity decided by the local oscillation frequency, which would be determined by a small variable RC circuit. The TDA7000 were no longer produced beginning in December 2003, although they may still be available on the market. There are also two later models, the TDA7010 and TDA7021. An example of the block diagrams of the TDA7010 is shown below in figure 4.2.2.1.4.

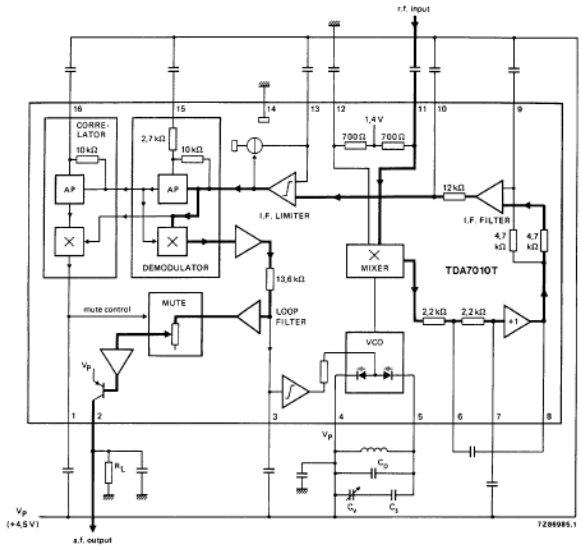


Figure 4.2.2.1.4: Block diagram of the TDA7010. Reprinted with permission from Digi-Key.

This circuit could be implemented on a PCB board more easily than the “do-it-yourself” method, but would eat a lot of space and use a lot of components. The variable RC circuit for tuning could also present some problems. One of the goals is for easy user frequency selectivity, and requiring the user to adjust a potentiometer would not qualify as such.

The second main option available would be to purchase a prefabricated chip that includes the FM receiver. These are generally digital circuits that can be controlled by a microcontroller to receive the desired station. Most of these circuits require external elements, although some are complete receivers in one integrated circuit. There are many different companies that offer varying solutions to this problem.

Sanyo is one of such companies. They have a line of FM receivers from the LV24000 series. These are all one-chip tuners, requiring minimal external components [10]. Some of these chips even include internal antennas. These tuners also have the ability to include RDS (Radio Data System). The RDS system allows for the output of the name of the station/song and band that is playing. One nice thing about this line of chips is their versatility; they have many different features available on different chips. Table 4.2.2.1.4 shows the different tuners in production and their feature sets, as they would be applicable to the needs of this project.

Model #	+AM Tuning	+RDS	I^2C Controlled	+Headphone Amp
LV24003				X
LV24010		X		
LV24100	X			
LV24230			X	
LV24233			X	X

Table 4.2.2.1.4: Sanyo FM tuner one-chip solutions [10].

Having the AM tuner would be a nice bonus, though it would not be necessary to this project. While any true commercial release of a project like this one would doubtlessly need it, FM is generally listened to much more often by most younger people. Especially since the function of our alarm is to wake the user up; a loud rock song would do the trick much faster than a talk show or news station (the kind of shows that are found more frequently on the AM band). The RDS decoder would be handy, though also most definitely not a necessity. The user knowing the name of the song that is playing is of less importance than the actual waking up. RDS functions would have more necessity in systems where the main function is to listen to and enjoy the music/programming. Most of the LV24000 lines of chips are 3-wire controlled, but the couple that is I^2C is noted. The headphone amp is not truly needed either, because another audio amplifier will most likely be used elsewhere in the GuSu system.

Sanyo was contacted via email, and they were very helpful. They provided help with locating the part in the event that the group decides to go with it, and they also gave important documents such as the datasheet, access to their LV24230 'Easy Radio' schematics and information, and offered to help however they could during the process.

Another company worth looking into is Silicon Labs. They also have many FM tuners on the market, most notably their line of Si470X ICs. These receivers do require external components, although the amount is minimal. Each model, like the Sanyo receivers, has different feature sets. The Si4702 and Si4703 models offer full FM receiving, with the Si4703 model including RDS decoding. The Si4704 and Si4705 models improve on the previous versions with Bluetooth capability, and will accept transmissions in that format. The 4705 chip also includes the RDS decoding. The Si4708 and Si4709 chips improve on those by decreasing the size to an amazing 6.25 mm^2, again with the Si4709 including the RDS decoding. An example of the Si4704/05 models (with necessary components included) is shown below in figure 4.2.2.1.5.

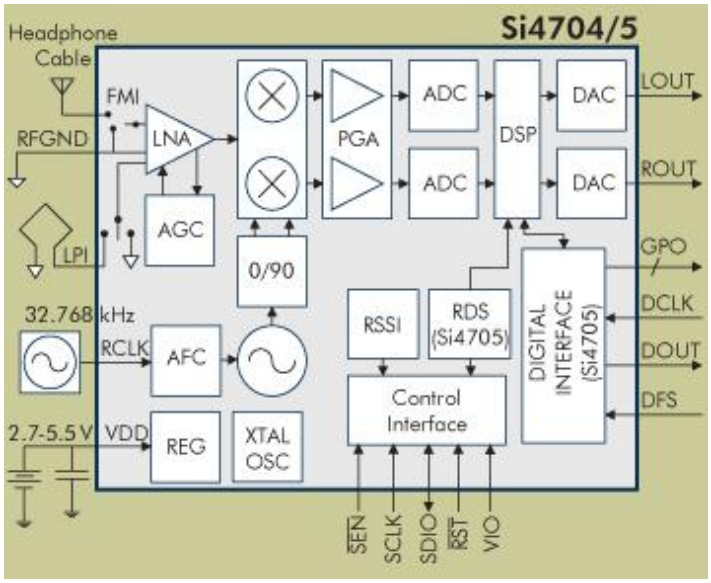


Figure 4.2.2.1.5: A block diagram of the Si4704/05 FM tuner. Notice the few needed external components. Reprinted with permission from Silicon Labs.

These models all seem to be difficult to order just one of; most of the authorized retailers require minimum purchases of over 2500 parts. Digi-Key had none available of any of the above models. Avnet Memec had only the Si4709 and the Si4703 part readily available, for $13.75 a piece. Nu Horizons had none of the above, and Mouser Electronics had the Si4704 for $9.21. The final part chosen, if any of the above, would depend on the characteristics given in the above paragraph.

Silicon Labs was similarly helpful when contacted about the group's project. They responded promptly to emails, and their local sales liaison quickly replied

with the datasheets, latest device errata, tips for PCB layout, and information on the embedded antenna.

A third option that could be possibly implemented would be from Silicon Labs as well. As of now, this project is not planning on supporting a USB reader. If that changes, Silicon Labs does manufacture an FM Tuner that is embedded into a USD card [11]. This looks to be used mainly for computer operating systems using their software to decode and play the music; however, it is reasonable to assume that it could be controlled with the microcontroller in this project. This would make it very simple to decide which audio output will be used as well.

4.2.2.2 Implementation Strategy

It was originally decided to use the Si4704 FM tuner. This was picked because it was cheap, small, and easily integrated and controlled. One was found on www.mouser.com for $9.21. It was ordered, it came in, and the group was very surprised as to how small it was (although it was exactly as advertised, 3mm X 3mm). It also came with no pins, and apparently needs a special machine to 'press' it onto a PCB. At this point, the Si4704 FM tuner was no longer a viable candidate for implementation. The Sanyo LV24000 series of FM Tuners was reinvestigated, and the same size/mounting issues remain with them. It was decided that the use of these specialty one-chip tuner chips should be left to the companies that have the abilities and the facilities to use them.

At that point, it was decided to use the TDA7000 chip. While all mainstream electronics stores (Mouser, Avnet, Digi-Key, etc.) were out of stock on it (if the part was even listed), a few of them were found on E-Bay. 2 were purchased for $21.25 from littlediode_usa, which surprisingly enough ships the product from the UK. Using resources found online, including the datasheet for it and a couple of helpful websites, the schematic in Figure 4.2.2.2.1 was created [12] [13].

As can be seen 4.2.2.2.1, there are many external components to be integrated, including 16 capacitors. The tuning is determined by three things: the input voltage, the variable coil (L1), and the potentiometer. The easiest way to set up the tuning for the user is to take the following steps [12]. First, for FM reception, the input voltage should be around 4.5V to 5V (5V will be used, as it is a common voltage level through the alarm clock). Next, the variable coil should be used to tune to the upper (or lower) frequency. In the case of FM reception, that would be 108 MHz (or 88 MHz). Finally, the potentiometer can be used to fine-tune the frequency between the specified ranges. This potentiometer also does not need to be attached directly to the PCB; it could be attached to the casing of the alarm clock with wires going to the PCB to be soldered on. This would allow the user to change radio stations without opening the casing.

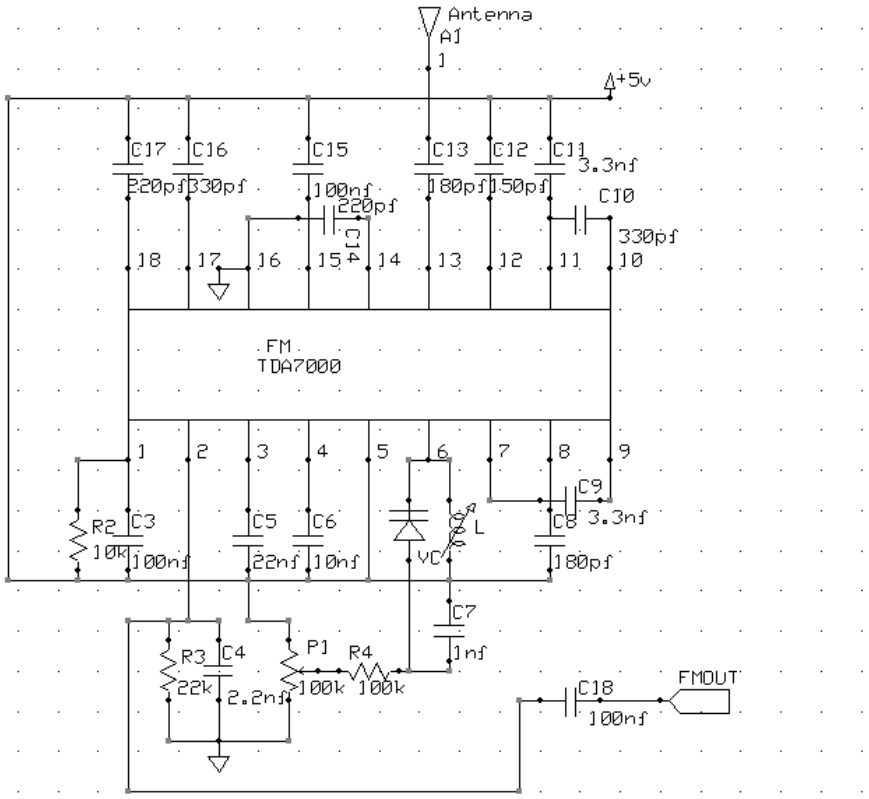


Figure 4.2.2.2.1: Final design of the TDA7000 FM Tuner implementation. Schematic created using ExpressSCH.

The antenna A1 can be built a couple of ways. Ideally, the clock will not have any exterior antennas. An interior antenna can be used that is at least 10 cm long, and to be on the safe side, a longer one will likely be used. Another important criterion would be a lack of interference, and one other component in the clock could cause some serious interference: the Zigbee device. Because of that, the FM tuner and the Zigbee device should be placed on opposite ends of the PCB, and the antenna for the tuner should stay on that end of the clock.

The audio output level of the TDA7000 is about 75mA. This could be greatly amplified by the audio amplifier, as needed to produce an optimal loud noise. The audio output signal would be routed to a multiplexer along with the mp3 decoder output. The microcontroller would determine which signal got passed through to the amplifier/speaker (if either).

4.2.2.3 Test Plan

The FM receiver would need to be tested separate from the other elements, with just the power supply connected to give the flat 5V input. The output should also be connected to the amplifier and the speaker, as hearing the radio (and the radio station's clarity, signal strength, etc.) is the only way to truly test the device. The most important thing to check would be that most of the common stations

that can be heard in a vehicle are audible with this device. If there are problems, the antenna could be lengthened to ensure that the problem is with the circuit or IC itself.

4.2.3 MP3 Decoder

To accommodate user comfort and and preference, the Get Up Stay Up (GuSu) Alarm System shall have the ability read and play MP3 files as audio output in addition to the other alarm audio/action options. This allows the user to choose any of music or sound they own or can create using one of the many available free and open source audio recording software as the alarm's wake up sound, greatly widening the alarm's range of available options for waking it's user. The storage of MP3 files is handled by the on board Secure Disk (SD) card slot discussed in its own section. In order to make use of the data stored on the SD card hardware and/or software is required in order to decode the files and process them. ID3 tags can also be handled to display the song name and artist on the GuSu's LCD screen. Whatever method is used must interface easily with the GuSu's micro-controller and sound output.

4.2.3.1 Research

Due to the GuSu's architecture a software solution to handling MP3 audio playing would require many hours of coding and pins to be dedicated to reading from the SD slot and passing along audio to the GuSu's speaker output, greatly decreasing the number of available pins for other uses. A hardware solution involving an MP3 decoder chip is a much more convenient and stable solution considering pins would only be required to send control signals to the MP3 decoder chip and relieve the engineers of the burden of software creation and testing. The team put a lot of effort into researching what MP3 Decoder to use with the GuSu prototype and it proved to be a difficult challenge deciding which decoder would work best with the design at hand. One of the main sources that were used in looking for MP3 decoders was Sparkfun electronics, this company seemed to hold a large array of different types of MP3 decoders. Some other companies found on the web that carried MP3 decoders were PJRC electronics and STMicroelectronics. The following contains information on the various MP3 decoders that were considered along with their specifications and why they were or were not chosen.

Possible solutions range from lone decoder chips, to integrate-ready boards, all the way up to fully assembled boards which only require a storage device be connected. While the third option is convenient, the prices range over a $100 and would be harder to interface with from the GuSu system without either modifying the board received or having separate controls. In addition the excess parts and components jeopardizes the design plans for keeping the GuSu small as possible. The second option does not have this problem, and are usually priced under $50. Below (table 4.2.3.1.1) is a table comparing some available MP3 mini circuit boards.

Item No./Name	On-board Chip	Interface	Output/Input	Dimensions	Price	Supplier
MINIMP3	VLSI VS1002	SPI	Microphone Audio Out	45 x 55 mm	$ 23.90	[15]
VS1003B-L SD Card Mini Player	VLSI VS1003B-L	SD card	Audio Out	N/A	80.00 €	[14]
VS1000B-L Tiny Player	VLSI VS1000B-L	USB	Audio Out	N/A	80.00 €	[14]
VS1053B-L	VLSI VS1053B-L	SD Card	Microphone Audio Out	N/A	80.00 €	[14]
MOD-MP3	VLSI VS1002	UEXT SD/MMC	Microphone Audio Out	57 x 45 mm	$ 41.94	[20]
BOB=07832	VLSI VS1002	UEXT SD/MMC	Microphone Audio Out	57 x 45 mm	$ 40.95	[21]

Table 4.2.3.1.1: Comparison of MP3 mini circuit boards

Some of these mini boards do include SD capability built in, however most use battery power, already have buttons soldered to the control lines, and include parts or capabilities not needed for the GuSu system. Another complication is controlling audio output for the system when the output lines are already connected to audio jacks and microphone lines for encoding. Extra hardware could present complications for running on battery backup during alarm time, and having unused hardware and controls sitting idle may also interfere with other signals to and from the main micro-controller. Therefore, with respect to routing of audio, control, power consumption, and overall design size, the optimum solution is to integrate a lone MP3 decoder chip.

One of the first decoders looked at was the VS1053b [20]. This chip was very interesting in that it had the ability to decode multiple audio file formats and not just MP3s. Some of the file formats the VS1053b is capable of decoding are Ogg, MP3, ACC, WMA, and MIDI. The team thought this chip might be a good fit for the GuSu design because of its potential to decipher the high quality Ogg sound format. Although this was a great attribute to this chip, the chip would require additional software plug-ins to handle these file formats. Some specifications for the VS1053b chip are that it can be controlled via a serial input bus and has the option of interfacing through uART for debugging purposes. The voltage supply to this MP3 decoder chip would be ran at 3.6V which was in the range of voltage that would needed to be used with the GuSu Design. Another key feature that caught the team's eye was the relative low power consumption at about 50mA. After realizing that SPI was the only interface with this MP3 decoder this solution was considered a poor product due to the amount of SPI pins that would be used with the ATMega644P microcontroller. Pricing on this chip was on average about $19.95 which wasn't that pricey for the high quality of this chip. Shown in figure 4.2.3.1.1 is a schematic of the VS1053b MP3 decoder showing the processing unit on the chip as well as the interfacing capabilities.

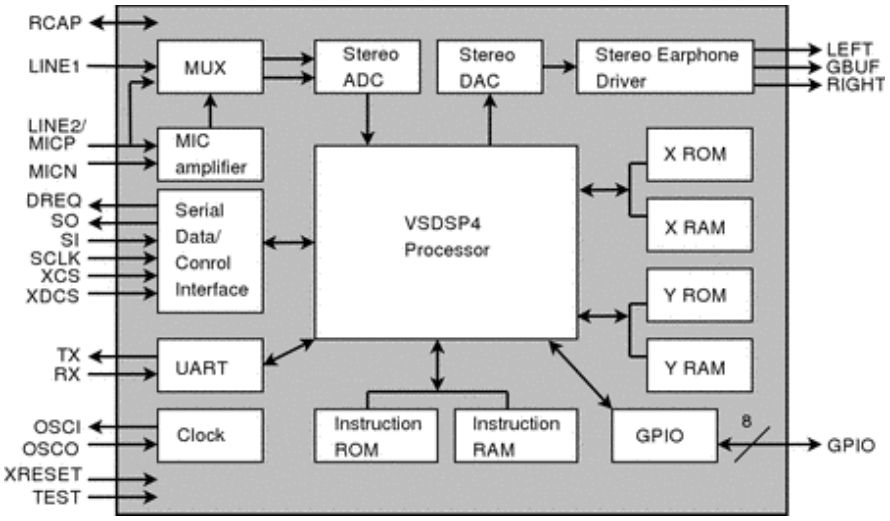


Figure 4.2.3.1.1: The package design of the VS1053b audio decoder. Reprinted with permission from VLSI Solution.

One other MP3 decoder that is manufactured from VLSI Solutions based in Finland, an integrated circuit developing company was the VS1001 [21]. This chip is also fabricated from the same company that makes the VS1053b. The VS1001 chip is not as involved and does not have all of the extra features as the VS1053b but it has some of its own advantages. The main difference in this chip is that it only supports the MPEG audio layer codec whereas the VS1053b has support for multiple audio formats. One other small difference is the voltage operating range which is considerably less than the 3.6V of the VS1053b; this chip will operate at about 3.3V which was an important factor in deciding which MP3 decoder to use. Some similarities of these two chips were that they both of them have the same clock operating frequency at about 13 MHz. The main interfacing solution to this MP3 decoder is through SPI, bearing in mind this chip does not have a uART interface to for debugging purposes. Pricing on the VS1001 averaged about $21.00; this was a bit pricey for this discontinued chip from VLSI Solutions. Once again, this solution only has SPI interface and was not ideal for the microcontroller the GuSu design uses due to the constraints of the other modules that would be using this interface.

Lastly, the team had found an ideal candidate MP3 decoder that could work easily with the ATMega644P microcontroller and the SD Card Reader. This chip is called the STA013 MP3 decoder developed by STMicroelectronics. The main feature of the STA013 decoder is that it can handle just deciphering of MPEG layer 3 formats, specifically MP3s. This differs from the other decoders that were researched in that the STA013 only deals with one type of file format making it idyllic for handling just MP3s which are the most popular form of sound format on the market today. When decoding these formats the STA013 has a special ability to output information in stereo, dual channel, or mono sound types. The STA013 can be run off of 3.3V helps with the GuSu design because this is low voltage for a decoder. Surprisingly enough the STA013 has very low power consumption at 85mW and also a lower clock frequency than the other MP3 decoders that were

researched running at 10 MHz. A key feature that caught the eyes of the team members was the I2C interface capability. Having essentially used all of the uART and SPI interface pins on the microcontroller there were a few digital pins left for I2C interface. Being that the STA013 would be the only component module that would be running on I2C this was a great addition to the GuSu development. Pricing on the STA013 was set at around an average of $20.00 which lined up with the other decoders looked at. Since all of the decoders were about the same price and due to the fact that the team needed a device with I2C operation the STA013 would be the right decoder for the price and features.

The STA013 has two modes of operation; it can work in either Multimedia mode or Broadcast mode. In the Multimedia operation mode the STA013 will decode the incoming MP3 data streams and output them to the speaker with much ease. This type of operating is controlled by buffer management that is embedded on the software within the decoder. While operating in the other mode, the STA013 will receive a data stream but has to check that the bit rate of the incoming MP3 data matches the actual bit rate of the stream that is being decoded. One important fact about the STA013 I2C data bus is that it will always act as a slave device instead of a master when connected to the microcontroller and speakers. Shown in figure 4.2.3.1.2 are the absolute maximum ratings of the STA013 decoder.

Symbol	Parameter	Value	Unit
V_{DD}	Power Supply	-0.3 to 4	V
V_i	Voltage on Input pins	-0.3 to V_{DD} +0.3	V
V_O	Voltage on output pins	-0.3 to V_{DD} +0.3	V
T_{stg}	Storage Temperature	-40 to +150	°C
T_{oper}	Operative ambient temp	-40 to +85 (*)	°C
T_j	Operating Junction Temperature	-40 to 125	°C

Figure 4.2.3.1.2: The absolute maximum ratings of the STA013 MP3 Decoder. Reprinted pending permission from STMicroelectronics.

Having done the research on these different decoders it seemed like the STA013 would be the best fit for the implementation of the GuSu system. The VS1053b seemed like a great choice the but team only cared about the MP3 formatting of audio files. Dealing with different formats of files would prove to complicate the design much more than it already is. Having the data interface as I2C also helped the team choose this chip over the others. Thus, this would prove to be the pick for the GuSu design. The implementation of the STA013 decoder chip will be discussed in further detail in section 4.2.3.2 of the document.

4.2.3.2 Implementation Strategy

After researching through the variety of different MP3 decoders it was decided that the STA013 MP3 decoder would be used. This section will discuss how the STA013 MP3 decoder will be implemented into the design of the GuSu prototype. Information that will be discussed includes examples of how the MP3

decoder will be connected and how it will work with the other modules on the GuSu prototype.

First off, the STA013 MP3 decoder will be running in Multimedia mode instead of Broadcast mode. This will ensure that the STA013 will automatically detect the bit rate of the MP3 data being passed through the chip. The STA013 will be able to receive data up to 20Mbit/sec and will detect the sampling rate of the MP3 data such as the MHz of each file. There has been some well known voltage issues with the STA013 when it is ran at 5V, so the team will most likely run the decoder at 3.3V to avoid these issues. Only 10 pins on the STA013 will be used. Two pins will ultimately be used for the I2C data interface; these pins will be connected to the ATMega644P microcontroller. Three of the pins will be used for the MP3 data source, since there is no good way of connecting the decoder to the SD Card without having some sort of control so this concern will need to be looked at when implementing the decoder with the SD Card and microcontroller. Four of the ten pins will be used for decoding the audio data and the last pin will be used for reset. Figure 4.2.3.2.1 shows a sample connection between the STA013 and the CS4334 digital to analog converter [16].

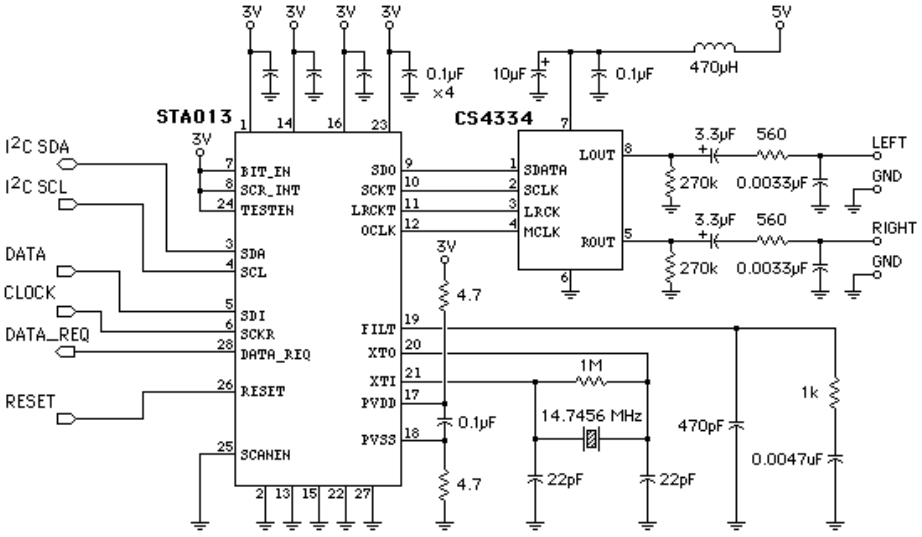


Figure 4.2.3.2.1: Schematic of the STA013 MP3 Decoder with the CS4334. Reprinted with permission from PJRC Electronic Projects.

Moving onto the actual connection to the microcontroller via the I2C data interface, one line will act as the data source to feed the input into the STA013 while the other line will act as the clock. There are four basic operations of the I2C interface when dealing with the STA013. One of the conditions is the start state, this will control whether to start or stop communication with the microcontroller. The other two main conditions that deal with data transfer are the send and receive bytes. The send byte will output the decoded MP3 data to the audio speakers while the receive byte condition will accept data pushed through

by the microcontroller to be decoded. Lastly there is a stop condition, when this is enabled the I2C bus will be set to idle mode and will not receive or transmit data.

Dealing with the software to control the MP3 decoder will be somewhat tricky. Software code will need to be implemented on the ATMega644P microcontroller; the first code will need to read in data from the SD Card. Once this has been completed the microcontroller should send data through itself and then to the STA013 MP3 decoder. There are two main functions that will need to be implemented with the decoder. These two functions will be the read and write operations. The write mode will write a certain types of bytes to a particular address on the decoder while the read mode is the opposite in reading data from those addresses. A configuration file is provided by STMicroelectronics that will help define the addresses that are used in the STA013 registers which were not previously defined in the data sheets provided by STMicroelectronics. The team will need to transmit this file to the MP3 decoder to update these addresses on the registers. After that matter in question is dealt with; the team will be mostly concerned about sending data to the decoder so it can output it to the audio speakers. Sending MP3 data to the decoder via the microcontroller seems easy enough. The decoder will not care about the MP3 bit rate of the files; it should use the data at a certain speed and then send a request signal when more data is needed. The request will be terminated once the STA013's buffers are full. One good feature about the STA013 is that it will ignore all other files except for MP3s and can even handle damaged MP3 files without corrupting the chip. One other thing to keep in mind is that the decoder cannot distinguish between different MP3 files, so software code will need to be implemented on the microcontroller to so that the decoder will be able to tell the difference between different audio files.

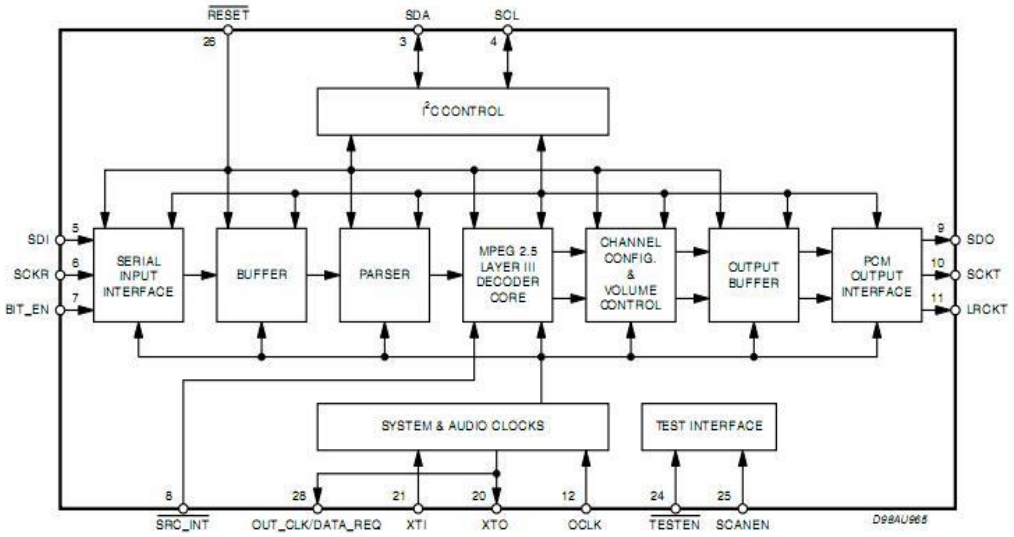


Figure 4.2.3.2.2: Block diagram of the STA013 hardware partitioning. Reprinted pending permission from STMicroelectronics.

Shown in figure 4.2.3.2.2 is a block diagram showing the STA013 MP3 decoder hardware partitioning. As seen in the diagram the decoder is controlled through the I2C interface. The data will be sent through the input buffer and then each subsequent package within the decoder. At the center of these stages the audio is decoded using the MPEG 2.5 Layer decoder and then passed to the output buffer where the data can be sent to the audio speakers.

Although the STA013 will not care about ID3 tags on the MP3s they will most likely be used to organize the MP3 files on the SD Card. The ID3 tags are data containers within the MP3 file format that can tell information about the audio files such as the artist name or song name along with other information. Most likely these tags will be used in junction with the LCD display. When songs are played from the SD Card the microcontroller will read the ID3 tags and display them to the LCD display where the user can read what song or audio file is playing. Overall there will be a lot of implementation to get the MP3 decoder to work jointly with the SD Card, the microcontroller, and the LCD display but it is a crucial part of the design for the GuSu prototype.

4.2.3.3 Test Plan

To test the functionality of the MP3 decoder the team will use a breadboard either owned by one of the team members or available in the UCF engineering laboratories. This device will be similar to the SD Card for testing purposes. This device can be tested alone to see if it is functional by connecting the voltage and the ground then testing the device with a multimeter. To check if the decoder is functioning correctly it will need to be tested along with the SD Card and the ATMega644P microcontroller. The team will most likely use a sample MP3 to send to the decoder to test the ability to decode the MP3. The audio speaker will need to be hooked up to the output of the decoder to check for sound output. If all of this works correctly the team will know that the decoder is working properly and can continue to implement the STA013 into the rest of the GuSu design.

4.2.4 Sim Card Reader

The purpose of the SD Card Reader will allow the user of the GuSu prototype to store their favorite songs and music on a widely available Secure Disk for playback through the GuSu's sound output. The reasoning behind this is so that besides the FM Tuner the user will also have a backup way of listening to their favorite music and not just their favorite radio stations. Using this method was found to be the most secure and most update to date. More will be discussed on how the SD card will be used and how it will allow the user to listen to their favorite MP3s.

4.2.4.1 Research

The first research that was done involved deciding what kind of storage device the GuSu design would use to pass MP3 files through the MP3 decoder and to

the speaker. There are many types of storage devices out on the market today. Some of the devices are Compact Flash, USB storage, hard drives and even SD cards. SD is a short acronym for Secure Digital which was developed by the Matsushita, SanDisk, and Toshiba companies. The purposes of these devices were to be used as portable solid state drives which could transfer pictures, videos, and other data. The main use of these storage devices were to be used in cameras, PDAs, and video game consoles to name a few. SD has been gaining momentum in recent years due to its relatively small size and large capacity compared to other storage mediums.

Card	Width, mm	Length, mm	Thickness, mm	Volume, mm^2	Mass, g
Compact Flash I	43.0	36.0	3.3	5108	3.3
Compact Flash II	43.0	36.0	5.0	7740	
Smart Media	37.0	45.0	0.76	1265	2.0
MMC, MMCplus	24.0	32.0	1.4	1075	1.3
RS-MMC, MMCmobile	24.0	16.0	1.4	538	1.3
MMCmicro	14.0	12.0	1.1	185	
Memory Stick Standard	21.5	50.0	2.8	3010	4.0
Memory Stick Duo	20.0	31.0	1.6	992	2.0
Memory Stick Micro	12.5	15.0	1.2	225	2.0
SD	24.0	32.0	2.1	1613	2.0
miniSD	20.0	21.5	1.4	602	1.0
microSD	15.0	11.0	1.0	165	0.27
xD	25.0	20.0	1.78	890	2.8
USB	Varies	Varies	Varies	Varies	Varies

Table 4.2.4.1.1: The relative sizes and masses of various multimedia storage devices [22].

After researching into the different types of storage devices through various websites the team decided the SD storage format would be the best fit for the design. Since all of these devices will be encased into the alarm clock a solution would be needed for a permanent storage device. USB would not be a good choice for this because the purpose of the USB sticks is to be plug and play as well as portable, they could also be flimsy inside of the alarm clock and maybe dislodge from its connector. A computer hard drive was looked into but this would also prove to be disadvantageous. A computer hard drive has moving parts and therefore the life expectancy would be less than that of the other solid state drives. A hard drive would also draw way too much power for the alarm clock to handle and interfacing these devices to the microcontroller or MP3 decoder would prove to be very challenging. After coming to the eliminating these devices from our design the team found that the SD storage medium would be the most advantageous for the GuSu alarm clock. Table 4.2.4.1.1 shows some different storage mediums and their relative size to one another. The actually storage capacity also varies from one to the other. Researching into the

Secure Digital format showed that this allowed for the most capacity variation between the different storage mediums.

The team's first choice when looking into SD storage formats was a special type of SD Card Reader that would interface with the MP3 decoder. Researching into GHIElectronics, a company specializing in storage mediums for embedded systems it was found that they produce an SD Card Reader that uses various interfacing connections such including uART, SPI, and I2C. This would prove useful in connecting to the MP3 decoder because this component contained all of the interfacing types.

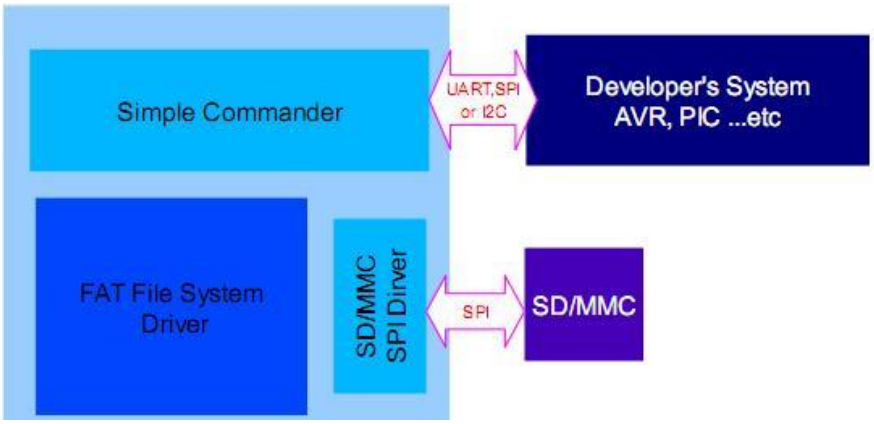


Figure 4.2.4.1.1: The diagram of the uALFAT and how it interfaces to other components. Reprinted with permission from GHI Electronics.

This device called the uALFAT [24] would provide support for the FAT16 and FAT32 file systems which is excellent when dealing with the MP3 format. This device would also allow the user up to four simultaneous file accesses and has a fast start up and reconnect. The average file read from this device is on average about 60KByes/sec. With GHIElectronic's new firmware the uALFAT would also be capable of interfacing with the new and improved SD cards like the SDHC cards. This Secure Digital High Capacity card allows for greater storage capacity. Some other key specifications are that it has a Real Clock Time and a low power consumption of about 12mA. This would prove useful with all of the devices that the alarm clock will be using. Lastly, all of the input and output pins will have a 5 volt tolerance which is in our range of 3.3V to 5V. Shown in Figure 4.2.4.1.2 is a sample image of the uALFAT device along with the pin schematic showing the different interfaces using uART, SPI, and I2C. The pricing was fairly expensive for this small component at $44.95 a piece from GHIElectronics. Due to the cost of this device more ways of implementing the SD reader would have to be looked into.

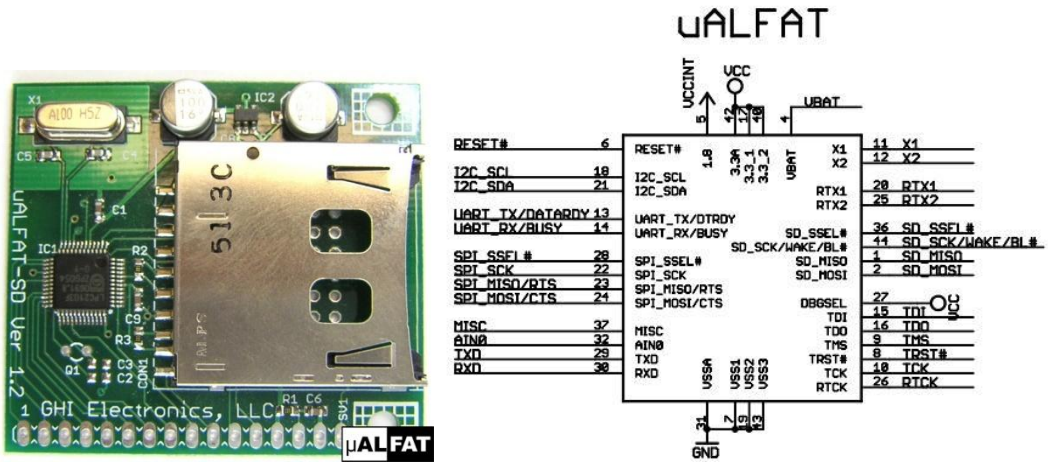


Figure 4.2.4.1.2: The uALFAT SD Card Reader as well as the pin assignments. Reprinted with permission from GHI Electronics.

Another type of SD Card Reader that was looked into was one that was found on Sparkfun electronics. This device called the Breakout Board [25] for SD/MMC devices is very similar to the uALFAT component that was researched above. There are small differences in this component such that the Breakout Board will only be using the SPI interface. After researching into which interface would suit best to work with the ATMega644P microcontroller the GuSu team found that the SPI interfacing would work best due to pin constraints on the microcontroller. The Breakout Board will also be using considerably less pins than the uALFAT. This component will also use a smaller voltage than the uALFAT at only 3.3 volts. Although this component seems like a great fit to work with the microcontroller the power consumption is a 0.5A which is much greater than the 12mA used by the uALFAT. Once again, power consumption considerations must be taken into effect when dealing with different components, the team must make sure that there is enough current to power all of the devices in the alarm clock. Figure 4.2.4.1.3 shows the Breakout Board as well as the PCB schematic of this design. This component also allows for data to be written and read from the SD card. The pricing of this module was only $17.95 and the team realized this would be a better buy due to the price as well as dealing with the amount of pins. The Breakout Board would be much easier to integrate to the MP3 decoder than the uALFAT.

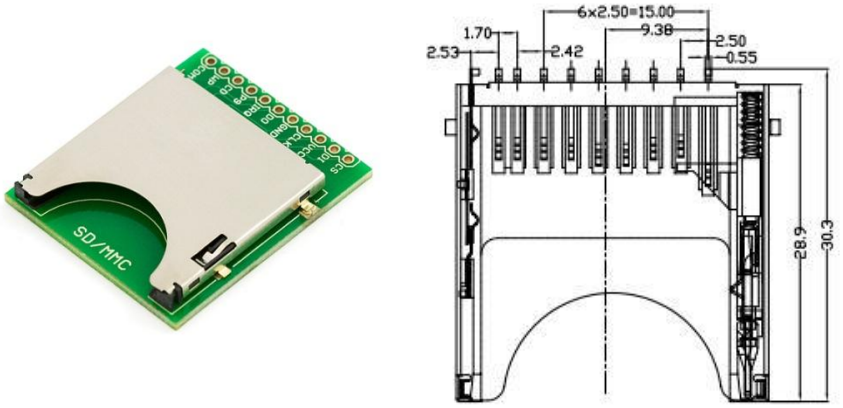
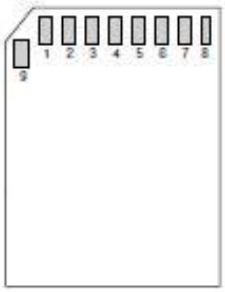


Figure 4.2.4.1.3: The Breakout Board SD Card Reader and its dimensions. Reprinted with permission from Sparkfun Electronics.

One more type of reader that was researched into was the USB SD Card Reader [26] from Sparkfun Electronics. This device seemed like a great choice for the GuSu prototype because it had support for all of the different types of SD formats such as SD, microSD, MMC, and compact flash. After realizing that the GuSu prototype would need some type of USB interface with the MP3 decoder the team thought that this could prove to be a challenging task and extra work. Finding a USB interface that will interface easily with the STA013 MP3 decoder would prove to add a lot of extra programming work on top of the large amount of programming already needed to be done. The team thought of another easy way to get around this by directly interfacing the USB SD Reader to the ATMega644P. Doing it this way could prove useful in that the microcontroller would control the data being read from the SD card and would pass it to the MP3 decoder and then out the speaker. The team finally decided that this would be the worst choice for the SD Card Reader because the USB interface to the microcontroller would need a way to be powered and this was adding to much implementation just for this one module of the GuSu alarm clock.

4.2.4.2 Implementation Strategy

After putting much research into the different types of ways to read data from a SD Card the team eventually found a very easy way to do this without the need for the need for the above card readers. The team found information on how to directly interface the SD Card to the ATMega644P microcontroller. Phil and Matt who were originally looking into the research and implementation of the SD Card Reader found that there was no way of directly interfacing the SD Card to the STA013 MP3 decoder without having some kind of control unit that would control the data flow being read from the SD Card. The team thought of adding a smaller microcontroller just for this purpose but this would seem like superfluous work to add another microcontroller just to read data from the SD Card and pass it to the MP3 decoder.



Pin	Name	Function (SD Mode)	Function (SPI Mode)
1	DAT3/CS	Data Line 3	Chip Select/Slave Select ($\overline{SS}$)
2	CMD/DI	Command Line	Master Out Slave In (MOSI)
3	VSS1	Ground	Ground
4	VDD	Supply Voltage	Supply Voltage
5	CLK	Clock	Clock (SCK)
6	VSS2	Ground	Ground
7	DAT0/DO	Data Line 0	Master In Slave Out (MISO)
8	DAT1/IRQ	Data Line 1	Unused or IRQ
9	DAT2/NC	Data Line 2	Unused

Figure 4.2.4.2.1: The SD Card pin numbers and their assigned functions in SD/SPI Mode. Reprinted with permission from interfacebus.com.

Figure 4.2.4.2.1 shows a standard SD Card with its subsequent pins; the table to the right shows what each pin is used for [27]. The method that would ultimately be implemented is directly connecting the pins of the SD card to the digital input lines of the microcontroller. The method of connecting the SD Card to the STA013 MP3 decoder was thrown out because of the need for a separate microcontroller to deal with data flowing out of the SD Card and into the decoder. The team is hoping that the ATMega644P will have enough processing power to deal with large data files such as MP3s; the concern is that there would be too much overhead for the microcontroller to deal with. Since this microcontroller will be controlling many other modules within the GuSu alarm clock there is a possibility the microcontroller could fail when reading data in from the SD Card. The team will need to do some testing before to make sure the microcontroller has the capabilities to handle this component.

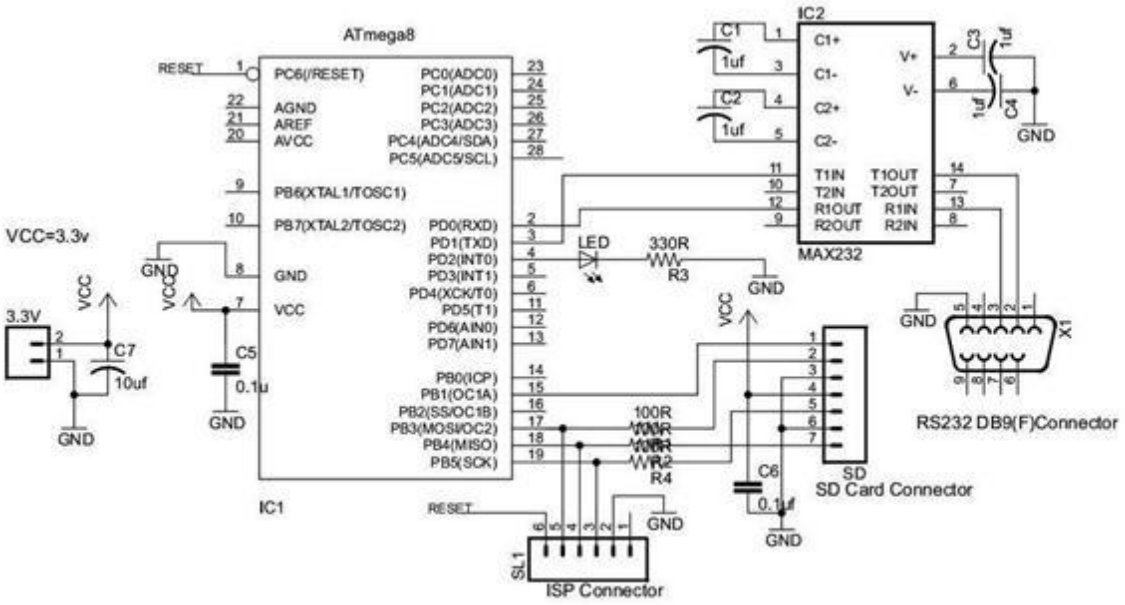


Figure 4.2.4.2.2: Schematic of the SD Card interfacing with the ATmega8 microcontroller. Reprinted with permission from dharmanitech.com.

Figure 4.2.4.2.2 shows a solution that was used for reading data from a SD Card with the ATMega8 microcontroller [27], this implementation will be very similar to the one that is used with the ATMega644P microcontroller. Chances are when the team implements the SD Card to the microcontroller resistors will have to be used when wiring the connections. The SD Card reader will also use a 3.3V source to enable the SD Card to work properly. The team will use standard electrical wires to and solder the pins on the SD Card to a digital input port on the ATMega644P microcontroller. Since the team is only concerned about reading data off of the SD Card the write pin will not be used. The team will be using a SD Card socket to input the storage device, pins are extended outwards from this socket and this is where the connections will be made by soldering the wires. The microcontroller will use three pins for the SD Card. One will be used for the chip enable, another for the clock, and lastly one digital input pin for parsing data to the microcontroller.

After connections are made to the microcontroller the team will need to format the SD Card so that the microcontroller will be able to detect it and read files. The standard format that will be used is FAT32 which is a Windows partitioning format used in many computers and storage devices. The team will then need to work on the code that will be used to control this device. The code will have to be implemented on the microcontroller since there are no other components connecting the two devices. The software implementation will most likely be coded in C or in the Arduino coding language. From here the microcontroller will send the inputted MP3 data through the STA013 MP3 decoder. The MP3 decoder will then output this through the speaker; more is discussed on the STA013 MP3 decoder in this document. This implementation will prove to be the most efficient because the data will be directly sent through the microcontroller. The research that was done in the previous sections showed the group that implementing the design this way would be much more trouble-free because there would be no second microcontroller to deal with sending data from the SD Card to the MP3 decoder.

4.2.4.3 Test Plan

To test whether or not the SD functionality is working the team will most likely be connecting the microcontroller into a breadboard and then wiring up the MP3 decoder and SD Card as well as the speaker. To make sure that everything is connected properly the team will use computer software to check the connections and the ultimate goal will be to see if the team can hear MP3s out of the speaker. This device will need to be tested along with the microcontroller and MP3 decoder to see if it is working properly whereas some of the other modules can be tested by themselves.

5.0 User Interface

5.1 Motivation

In designing a user interface assumptions must be made and stuck to about the environment and user. An assumption that will affect most of the assumptions to follow is that the system will most likely be interacted with between dusk and dawn. For the environment there is a bedroom, likely low lit, and the GuSu device positioned within arm's reach from the bed. For the user it is assumed they are not fully awake, either due to being sleepy or having just been awoken by the system. It is also assumed that the user is a legal adult and has had interactions with basic consumer electronics and alarm clocks.

Keeping these assumptions in mind the more basic and default options given to the user should function as the user expects them to rather than a method that is easier to implement, or seems more sensible. Designs on paper can often seem perfectly intuitive and even innovative, but once realized can be clumsy or frustrating. In the design of the GuSu's overall user interface the combined graphical user interface and physical user interface are built around these ideals with the previously mentioned assumptions to create an interface that is simple to use, and straightforward as possible, hopefully so much so that almost anyone can use the GuSu system without reading an instruction manual. Before the UI is laid out in detail, figure 5.1.1 presents a block diagram of the UI's finalized abstract components.

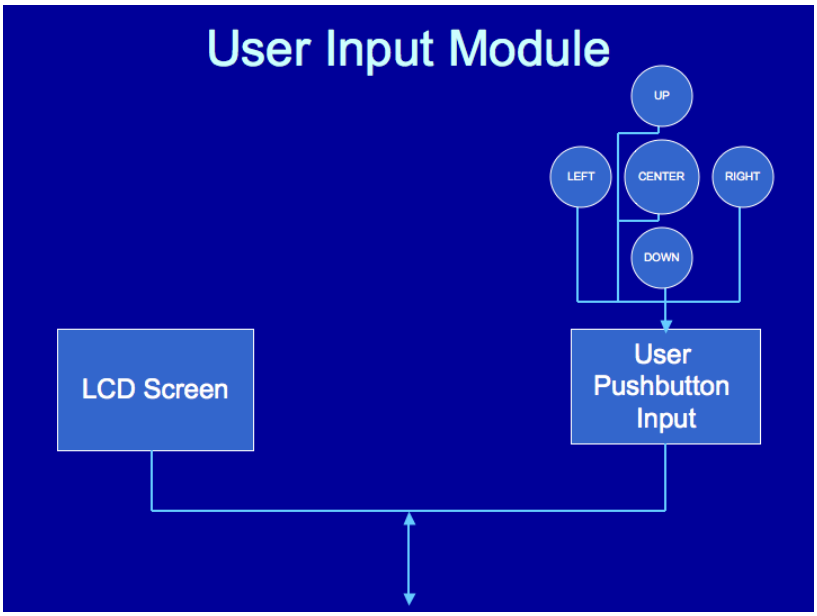


Figure 5.1.1 User Interface Block Diagram

5.2 Liquid Crystal Display

The alarm clock will also be integrated with a LCD display for easy user access. LCD is short for Liquid Crystal Display and will be defined more in detail in section 5.2.1 of this document. The LCD display will be integrated into the alarm clock through the microcontroller that has been chosen in this design. The main purpose of having this display is so that the user will have an easy way to access the controls of the alarm clock. Some of the common features that will be used on this display are a friendly user control panel as well as some buttons. The user control panel will allow whoever is using the alarm clock to set their time, alarm time, and other features that will be discussed in detail. The buttons will provide support to the user to change settings within the control panel. A discussion behind the history of LCD displays and the research that was put into the design will be examined in section 5.2.1 of the document.

5.2.1 Research

First off, a discussion of a brief history of the Liquid Crystal Display will be explained. Following this discussion will be important information about how this specific type of LCD display was chosen along with the research that went along with this decision.

Back in the late 1800s a chemist by the name of Friedrich Reinitzer had found liquid crystals in the cholesterol molecules of carrots. In the early 1900s there was various research done on these liquid crystals, they were finally classified into three types which were called nematics, smectics, and cholesterics. By the time of 1962 a member of the electronics company RCA named Richard Williams was the first to notice that the liquid crystals had electrical and optical properties by applying a certain voltage to them. In the following decades more work was done by researchers to harness the practical value of these crystals and how they could be used in electronics in modern society. In the early 1970s various patents were made in the USA as well as in Europe for creating the first twisted nematic LCD display [29]. Twisted nematic displays mean that display contains liquid crystals that will twist and untwist at different angles which allow light to pass through them. Once this happened the dominos fell and work on the LCD displays increased dramatically with more and more patents being made in the following years. Figure 5.2.1.1 shows a common active-matrix LCD display. The liquid crystals are encased within two glass plates which have horizontal and vertical filters on either side of the glass. These are what create the horizontal and vertical sync lines on your LCD monitors or displays. To add color to the screen a color filter is added over the horizontal filter.

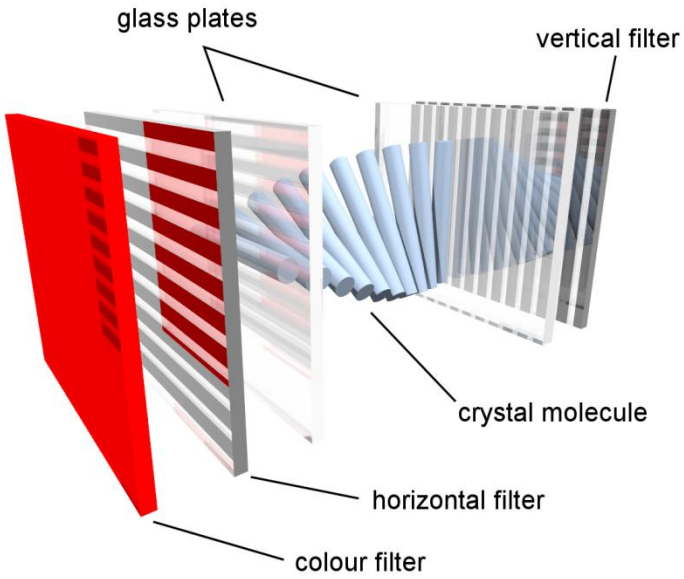


Figure 5.2.1.1: The common components of a LCD display and how it is assembled. Reprinted with permission under the General Public License.

By the time the new millennium had hit LCD displays were being sold more than CRT displays. CRT is short for Cathode Ray Tube. These are usually the more bulky monitors or displays that are used with computers, CRT displays use a vacuum tube with three separate electron guns that shoot electrons to the screen. These electrons will emit from a screen that will show images on the display. By present day the LCD displays are the most popular form of imaging ranging from computer screens, laptop screens, and now more than ever TV screens.

Now, continuing onto the research methods that were used in choosing our specific LCD display. The team had researched various displays throughout the internet trying to find a LCD screen that would work with our microcontroller. Some issues in finding the perfect screen were backlighting, viewing angle, power consumption, and whether or not the display was capable of displaying characters. Detailed in the next paragraph is a comparison of two different LCD displays that were considered before choosing the appropriate one. Ultimately, the uOLED-160-G1 would prevail over the other displays that were looked into.
The first LCD display that was looked at was the 16x2 character LCD RS232-TTL. This display had many of the features that were required by our alarm clock module. This display required a +5V supply to power it while also having a high operating temperature range of up to 150 degrees Fahrenheit. To work with the specific microcontroller that was chosen the display had to have an SPI interface. SPI is short for Serial Peripheral Interface Bus, which is a synchronous serial data link that will operate in full duplex mode. Some advantages of the SPI interface over other interfaces such as I2C or SMBus are that it has a higher throughput as well as simple hardware interfacing. The only real disadvantage of

the SPI interface is that it requires many more pins to run than the other interfaces. Pricing for this display was set at about $24.00 which wasn't bad for this type of display but was a bit pricey. After coming to the conclusion that the LCD RS232-TTL wasn't a suitable fit for the design because of limited capabilities of the LCD display and also the user friendliness of the hardware, the uOLED-160-G1 seemed to be the right choice for the design that was chosen.

The team decided that the just standalone character LCD displays wouldn't look aesthetic enough for what the GuSu design was trying to accomplish. The team decided that the OLED graphic displays would be a great visual aide for the alarm clock. Another display looked into from the same company that makes the uOLED-160-G1 was the uOLED-320XX-P1. Some of the specifications about this display were that it had a 240x320 pixel resolution with 256, 65k, or 262k true life colors on the AMOLED screen. The viewing angel was near perfect allowing the user to see the screen from about 180 degrees. The interface was SPI with the same pin configuration as the uOLED-160-G1 graphics display. The voltage supply could be run at 3.6V to 5.5V with the current being on average at 60mA which was pretty minimal for this type of display. Some extra features that were considered were the addition of the microSD memory card adapter and an audio amplifier with an 8 ohm speaker for audio playback. This display was great for the GuSu design but at a costly $149.95 this was definitely out of the price range for the GuSu team so this display was eventually ruled out.

Now, moving onto the uOLED-160-G1 graphics display and why it was chosen. This display was suited well for interfacing with the microcontroller having an uncomplicated and straight forward approach to connecting it to the microcontroller. To understand the properties behind the OLED display a comparison will be made between this display and the standard TFT display. OLED stands for Organic Light-Emitting Diode while TFT stands for Thin Film Transistor. Most LCD displays that are seen in stores and used with computers are TFT displays, but the OLED display has many advantages over the TFT. To understand why the OLED works the way it does a figure is provided to help the reader understand why it works. Organic material is placed between an anode and a cathode which is composed of electric conductive materials. By applying a voltage through the organic materials holes and electrons are injected to and from the anode and cathode in which luminescence will occur. To understand why the OLED is a better fit an understanding of TFT displays must be discussed. The TFT is a special kind of field effect transistor in which films of a specific type of semiconductor are used as both the active and dielectric layers sitting upon a substrate usually made of glass.

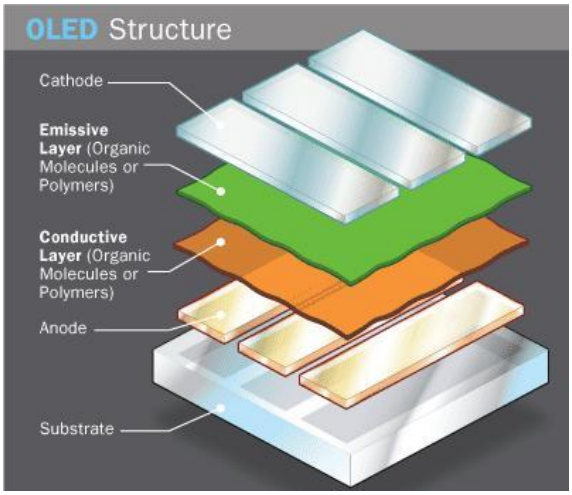


Figure 5.2.1.2: The hardware structure of an OLED display. Reprinted pending permission from HowStuffWorks.com.

Figure 5.2.1.2 shows the structure of an OLED display. This display was suited well for interfacing with the microcontroller having an uncomplicated and straight forward approach to connecting it to the microcontroller. There are many advantages that the OLED display has over the TFT and those will be discussed briefly. One of the major advantages is the high contrast ratio which can be up to 20 times the amount as a normal TFT display. The purpose of the contrast ratio is that it is a ratio of the luminance of the bright color being white to the darkest color being black. A higher contrast ratio will provide a more detailed and fine picture than something with a low ratio. Another great benefit is the response time of the OLED screen. These displays can have an astonishing 50 microsecond response time while the normal TFT LCDs have somewhere in the range of 3000 to 30000 microseconds [30].

Power consumption is also considerably less than TFT because in the TFT displays a backlight must always be on to show images on the display while the OLED uses a black background to minimize power utilization. Knowing these facts about power consumption will also help with the changing economies and the green movement. With the world's energy crisis at hand more and more people are turning to energy efficient products and the OLED provides that over the TFT displays.

Now, even with all of these advantages there are also disadvantages that come with them. The major one would be the price, since the OLED is more expensive to manufacture it is obvious that these displays will cost more. The other drawback being the display's lifetime, the average lifetime of an OLED display is about 20000 hours whereas the TFT displays get about 50000 hours of use. Shown in figure 5.2.1.3 are sample pictures of the uOLED-160-G1. The left image shows the display sitting on top of the chip while the right image is the back of the OLED LCD display. The GOLDELOX-GFX graphics processor

situated in the middle of the chip, it is a black square. The graphics processing unit will be defined in further detail in section 5.2.3 of the document.

Figure 5.2.1.3: The µOLED-160-G1 LCD display as well as circuit board it is connected to. Reprinted with permission from 4D Systems.

5.2.2 Implementation Strategy

The alarm clock will also be integrated with a LCD display for easy user access. The LCD display will be integrated into the alarm clock through the microcontroller that has been chosen in this design. The main purpose of having this display is so that the user will have an easy way to access the controls of the alarm clock. Some of the common features that will be used on this display are a friendly user control panel as well as some buttons. The user control panel will allow whoever is using the alarm clock to set their time, alarm time, and other features that will be discussed in detail. The buttons will provide support to the user to change settings within the control panel. As seen in Figure 5.2.1.3 the SD card will be inserted at the back and base of the LCD display. Powering the uOLED-160-G1 will be the GOLDELOX-GFX graphics processor; this module will be discussed in further detail in section 5.2.3 of the document. Also shown in Figure 5.2.2.2 are the dimensions of the uOLED-160-G1 LCD display with a diagonal length of just about 1.7 inches. The cost of this display was set at around 100$ which is pretty pricey but not out of the teams budget.

Some of the specifications of the uOLED-160-G1 LCD display are as follows. The display will have a native resolution of 160x128 pixels and 256/65K true color. This screen does not have any backlighting which is ideal for what this design is trying to accomplish; the imagery is near perfect having a 180 degree viewing angle. The uOLED-160-G1 will use a voltage supply from 3.6V to 6V. Although the standard that will be used will either be 3.6V or 5V depending on whether or not the SD memory card will be used along with the display. The micro SD memory card will allow for images or animations to be displayed on the LCD, allowing a range of 64Mb to 2 GB memory cards. To connect this device to the microcontroller that was chosen a simple 5 pin interface is included on the

LCD display. These 5 pins account for the VCC, TX, RX, GND, and RESET connections and will be connected to the microcontroller via SPI interface.

Interfacing of the uOLED-160-G1 to the microcontroller will be pretty straight forward. The display will be connected via the SPI interface and five pins will be used on the LCD to power and connect this device. VCC will be set in the range of 3.6V to 5V to power the display. The TX and RX which will transmit and receive connections on the LCD display will allow the device to communicate with the microcontroller. These pin connections will be used when sending signals and data to the display from the microcontroller. The microcontroller will also have the ability to read information off the LCD. The software code to implement this will be discussed in the Software section of the document; this will include the initial pseudo code. The last pins to be connected will be the ground and reset. Reset will be used to change the device back to the factory defaults.

This display will fit snugly onto the GuSu prototype case. The uOLED-160-G1 will be facing on the front side of the case and this is where all interaction between the GuSu prototype and the user will occur. Pushbuttons accompany the LCD display which will allow for configuration changes. Seen in figure 5.2.2.1 is a sample of the connections being made between the uOLED-160-G1 and a microcontroller using the 5V system.

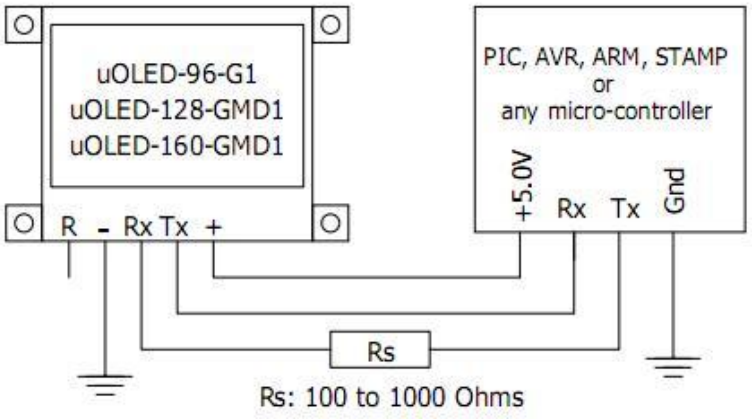


Figure 5.2.2.2: The interface between the uOLED-160-G1 and a microcontroller. Reprinted with permission from 4D Systems.

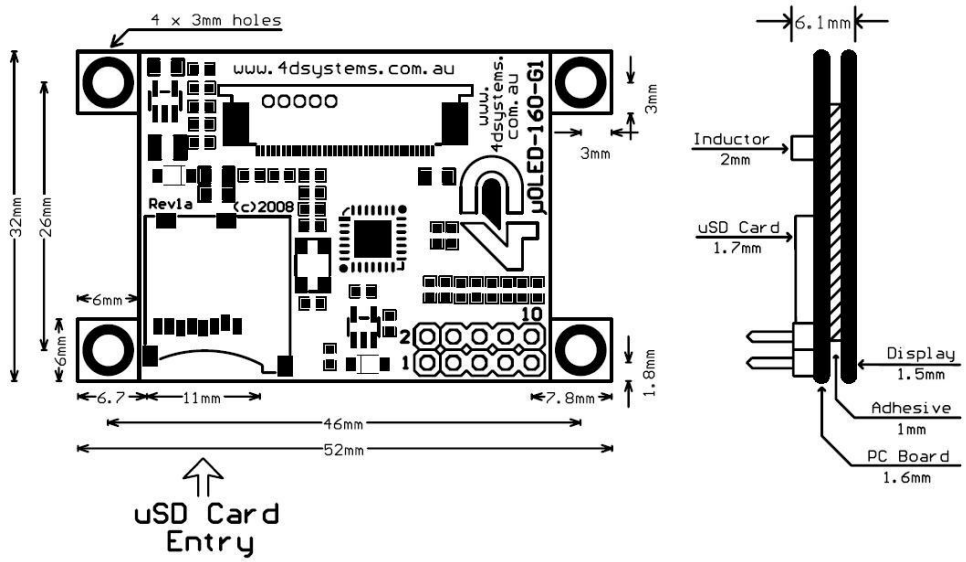


Figure 5.2.2.2: Layout of the uOLED-160-G1 LCD display and its dimensions. Reprinted with permission from 4D Systems.

5.2.3 LCD Driver Chip

In this section of the document the LCD driver chip will be explained in detail. This is the graphics processing chip that is located in the center of the LCD display chip. This particular chip is called the GOLDELOX-GFX graphics processor [30]. This type of processor will easily interface with many of the different types of TFT and OLED displays. The graphics processing unit will allow an output of high-level graphics along with multiple I/O functions. All of these functions are controlled by E.V.E. which is short for Extensible Virtual Engine. This new type of virtual processor created by 4D Labs will allow code that was implemented to work with the GOLDELOX-GFX processor to work on other 4D Labs processors. This is a major advantage to this new type of architecture because once you have implemented code to work on the GOLDELOX-GFX you will not need to change the code to work on other devices. 4D Labs has written software named the 4DGL Workshop as well as the Graphics Composer that will allow the software developers to easily code new features for the GOLDELOX-GFX and the OLED displays. There are several features that will interact with EVE such as SD memory card drivers, graphic functions, flash memory, and eventually an output via SPI interfacing. The GOLDELOX-GFX requires 4DGL which is a high level graphics language required by this processor. Some of the features that this graphics processor is capable of handling are color images, animations, and even video clips. Even though for this design's implementation these advanced features will probably not be used that much. Seen in figure 5.2.3.1 is the schematic of the GOLDELOX-GFX processor. Since this design focuses on the OLED display rather than the graphics driver the pin assignments of the processor will not be discussed in further detail. Shown in figure 5.2.3.2 are the absolute maximum ratings about the GOLDELOX-GFX processor.

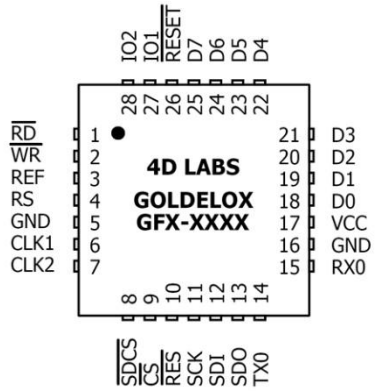


Figure 5.2.3.1: Package and pin assignments of the GOLDELOX-GFX graphics processor. Reprinted with permission from 4D Systems.

Absolute Maximum Ratings	
Operating ambient temperature	-40°C to +100°C
Storage temperature	-65°C +150°C
Voltage on any digital input pin with respect to GND	-0.3V to 6.0V
Voltage on analog pin with respect to GND	-0.3V to 6.0V
Voltage on VCC with respect to GND	-0.3V to 4.0V
Maximum current out of GND pin	300mA
Maximum current into VCC pin	250mA
Maximum output current sunk/sourced by IO1 or IO2 pin	4.0mA
Total power dissipation	1.0W

Figure 5.2.3.2: The absolute maximum ratings of the GOLDELOX-GFX graphics processor. Reprinted with permission from 4D Systems.

5.2.4 Test Plan

To test the effectiveness of the LCD display there will be multiple things that need to be tested. First off, to test if the LCD display has arrived in tact the team will supply a voltage and ground to the display and turn the power on. If this succeeds then the display will be known to work. To actually get any images and a menu on the LCD display more work will be required to connect the display to the microcontroller. The team will be using a breadboard to connect both of these devices and the software code that the team has written will need to be implemented onto the microcontroller. The team can also use the built in SD Card reader on the uOLED-160-G1 to test the displays quality when dealing with images or animations. The team will first off test if the display can communicate with the microcontroller and the team will send some commands to the display such as a simple time output. If this proves successful the team will then work on implementing the rest of the features that the LCD display will use. Some of these features will be a menu system for the user so they can change settings and also a way to show what each push button will do. After the basic settings are tested the team can then further apply the rest of the components that deal with the display such as music being played off of the SD Card.

5.3 Physical User Interface

Since the PUI greatly influences the design and capabilities of the GUI, its overall architecture must be decided on first. Options vary from two pushbuttons to a full thirty-six button keypad, and these can be bought pre-labeled, or be positioned and labeled on the device. Building a custom button system however would require the user to do more learning in order to use the system and at times force them to look at the buttons when interacting with it. Furthermore it would create additional engineering hours for the design and implementation of such a button display, making this option the least desirable.

5.3.1 User's Expectation

In order to design a physical interface that the user will need little to no instructions for a variety of alarm clocks on the market were looked at. Rising in popularity are alarm clocks that can both charge and redirect audio from iPods. These devices make a good competitor model to the GuSu system as they tackle the challenge of allowing a user to have a variety of options, including personalized audio selection, for waking. The latest model in the iHome lineup, the iP9SR, has the features listed in figure 5.3.1.1 f33].

- Wake or sleep to iPhone, iPod®, AM/FM radio or buzzer
- Universal dock with inserts to fit docking iPod models
- Weekday/weekend alarm settings
- Programmable snooze times
- Charges iPhone or iPod while docked
- High-fidelity stereo drivers in specially designed Reson8® speaker chambers deliver astounding clarity, depth, and power
- Line-in jack
- Gradual Wake and Gradual Sleep increase/decrease Alarm/Sleep volume so as not to startle the user
- Full function remote control controls unit and iPhone/iPod menu functions
- Dual alarm with AM/FM presets
- Bass, treble, 3D and balance controls for best sound
- Extra-large, backlit custom LCD Display with dimmer
- DST switch for quick daylight-savings time adjustment
- Backlit buttons

Figure 5.3.1.1: List of features for an iHome

Most of these features work towards offering the user a wide selection of options while making control of the device straightforward. Widely recognizable symbols are used where ever possible to mark control buttons (such as the power, play/pause, and alarm 1/2 buttons), scroll wheels with a center buttons which mimic the iPod's control interface are used for volume and numerical incrementing and decrementing, and a large button that is as wide as the control area doubles as a snooze button and LCD back light dimmer. The GuSu alarm system will take advantage of the computing power of its micro controller to have

seven day programmable alarm times. In a similar fashion the iHome takes advantage of what is now cheaply available electronics to give the user weekday/weekend alarm timing, this also gives weight to the design assumption made for the GuSu system that more and more users have use of such a capability. College students especially have varying morning schedules that often fluctuate day to day and many times over a year.

An example of a product on the market which solves the other challenge the GuSu tackles, waking a user who needs a little more motivation than the alarm simply sounding, is the Flying Alarm Clock from Vat19.com. This alarm clock features a small plastic propeller that is launched at alarm time that the user must retrieve and replace on top of the clock to silence the alarm. For many this is ample extra action needed to awake and stay out of bed, however, there is nothing preventing the propeller from landing on the bed or within arm's reach of the bed, allowing the user to silence the alarm while remaining in bed. Disregarding this possibility, it is unlikely it will take much time to retrieve the propeller and this violates one of the key design requirements of the GuSu alarm system, the user must not be allowed to occupy the bed during the alarm time. The Flying Alarm Clock from Vat19.com also has a snooze switch and modes that do not utilize the propeller, violating another requirement of the GuSu system that the only way to silence the alarm in a timely fashion is to leave the bed [34].

Most of these products such as the iHome and iLuv have a simple backlit LCD screen, basic play/skip/stop controls for the attached mp3 device, and just enough buttons for adjusting clock and alarm times. The GuSu alarm system handles mp3 audio through the SD card slot and MP3 decoding module discussed in their sections of this document, but still requires a method for time setting. Most digital display alarm clocks use similar control methods as the iHome and iLuv involving holding down a button for either clock time or alarm time and tapping or holding a button for hours or minutes which increments the desired value. In table 5.3.1.2 various available pushbuttons and keypads are compared for price and function.

Item No.	**Description**	**Supplier**	**Unit Price ($)**
PSPST01	SPST off-on Red Pushbutton	[31]	0.45
PSPST10	SPST on-off Black Pushbutton	[31]	0.45
PRBLK	SPST Round off-on Black Pushbutton	[31]	0.65
SPBLK	SPST Square on-off Black Pushbutton	[31]	0.45
PLBLK	Black Large Pushbutton	[31]	0.45
KEYSWITCH01	Mini SPST Key Switch	[31]	4.90
KEYPADSM	12 Button Small Keypad Switch, 51mm Wide, 64mm High, 7mm thickness	[31]	2.20

KEYPAD4X4B	16 button keypad switch, 75mm wide, 80mm high, 17mm thickness	[31]	4.40
(SWP) 3W720-SS20	General instrument snap action pushbutton SPST, red lens, can be lighted, 5/16" mount, 1-1/16" long.	[32]	4.00
(SWP) SPM	2 pole 5 amp contact pushbutton switch, maintained contact	[32]	9.00
GH5008-ND	KEYPAD 12 KEY BLK FRONT PNL MNT	[33]	12.26
MGR1104-ND	KEYPAD 36 KEY W/LEGEND SEALED	[33]	75.10
MGR1501-ND	KEYPAD 12 KEY VANDAL RESISTANT	[33]	64.72
MGR1516-ND	KEYPAD INDUSTRIAL 4 KEY ARROW	[33]	30.86
GH5010-ND	KEYPAD 16 KEY BLK FRONT PNL MNT	[33]	13.66

Table 5.3.1.2: Table comparing basic options for pushbuttons and keypads

A full thirty-six button keyboard is unnecessary for this application, and given the assumptions about the user and environment, high end push buttons are also unnecessary. From the user's perspective the 4X3 or 4X4 matrix keypads would seem an easy choice. Setting the various time based settings would be simple, menus and non on/off options could be itemized and navigated quickly. However, the user would at times need to look away from the GUI to examine the keys to ensure proper input, and pay attention to what each button does as it changes with menu navigation. Using number input and having so many keys would also require invalid input handling, while controls for number incrementing would only require having loops for hours and minutes that go from 1 - 12 and 0 - 59 respectively.

For the GuSu alarm system the GUI should provide the user with detailed information and allow them to interact and control the device with minimal action on their part. So the entire user interface will be built on and designed around a moderately sized graphical liquid crystal display and five pushbuttons oriented in an UP, DOWN, LEFT, RIGHT, CENTER orientation. While this will require more programming and testing during the design and implementation phase, it provides the user with the aforementioned easy interaction, spending more engineer hours rather than forcing the user to do more to use the system.

5.3.2 Key switch

As a deterrent to silencing of the alarm, the idea of using a key switch on the bottom of the GuSu unit for changing between RUN and SET mode was proposed. This would force the user to have the key nearby, turn the unit over or on its side, and insert the key to turn the alarm from RUN to SET. This would be easy to implement, and relatively effective since the key could not simply be left in the key-turn slot indefinitely. Unfortunately this not only gives the user a

relatively quick way of silencing the alarm and returning to bed, but if the key is lost or damaged the user loses control of the GuSu system. Additionally having to keep track of the key and turn the unit over each time they wish to change modes would become a hassle after several uses. Therefore this idea is being left out of the design due to the above complications and negative aspects.

5.3.3 Physical User Interface Implementation

The area surrounding the push buttons shall be flat, allowing the user to find the buttons easily by brushing their fingers across the top surface of the GuSu unit. The CENTER pushbutton should be slightly lower than other pushbuttons, making it easier for a user to orient their fingers. All other pushbuttons (UP, DOWN, LEFT, RIGHT) shall be the same height and oriented at ninety degree intervals around the CENTER pushbutton. LEFT and RIGHT shall be aligned along a line parallel to the front of the GuSu unit, while UP and DOWN are oriented on a perpendicular line to LEFT and RIGHT. Going clock wise the directional pushbuttons shall be wired in the following order; UP, RIGHT, DOWN, LEFT.

In the interest of ease of programming the GuSu system will be built using off-on pushbuttons, and for now will be designed around the SPST round off-on black pushbuttons available from Futurlec. These have a mounting diameter of 10mm and height of 20.5mm. Figure 5.3.2.1 is a stock photograph of the pushbutton followed by theoretical schematics of the pushbutton layout.

Figure 5.3.2.1 Stock Photo of Pushbutton
Reprinted with permission from Futurlec.

5.3.4 Physical User Interface Test Plan

A small program shall be written to run on the micro-controller which gives simple LCD outputs indicating what logical direction (UP, DOWN, LEFT, RIGHT, CENTER) is being interpreted from pushbutton signals to ensure that pin assignment and circuitry is correct for all five pushbuttons. If delays occur with getting the LCD unit fully functional an LED can be wired to a spare pin on the micro-controller and programmed to use either blinks or light up durations to indicate which pushbutton has been pressed.

5.4 Graphical User Interface

The graphical user interface will be run by software downloaded to the on board micro controller. It will be displayed to the user on the LCD screen that has been chosen, and controlled by the pushbuttons discussed earlier. For the five push buttons to be able to control everything necessary for daily operation the GUI will emulate a branch model for menu handling. That is, upon entering the SET mode the user will be presented with a list of categories to set, and upon entering one of these menus will be presented with a sub set within that menu, and enter a sub set of each of those options, and so forth. While this is not the quickest or simplest design, it is straightforward, can be easily navigated, and would only require at most two buttons for changing/setting various features, options, and values available as user preferences.

The branch menu model was chosen for its common implementation and intuitive use. It is used in many retail electronics such as camcorders, Texas Instruments calculators, iPods, and more. Branch menus also relieve the user of needing to pay much attention to the pushbuttons other than knowing that their fingers are placed correctly over them, allowing them to keep their focus on the screen once they have begun interacting with the device. Even if the user has no prior experience with branch menus they should be able to quickly figure out how to do anything they intend as far as choosing and adjusting settings for the GuSu system. Furthermore, as mentioned earlier, branch menus greatly reduce the amount of invalid input checking necessary, as it gives the programmer full control over deciding the entire range of possible input. The sections immediately following this discuss the abstract design of the graphical user interface and how the GUI and PUI interact.

The graphical display will have two modes; RUN and SET. The names are self-explanatory; however figures 5.4.1 and 5.4.2 show an hypothetical layout and list of what will be shown when the GuSu is in RUN mode respectively. Figures 5.4.3 and 5.4.4 present a hypothetical layout for a setting screen and a hierarchical list of the SET mode's branch menus.

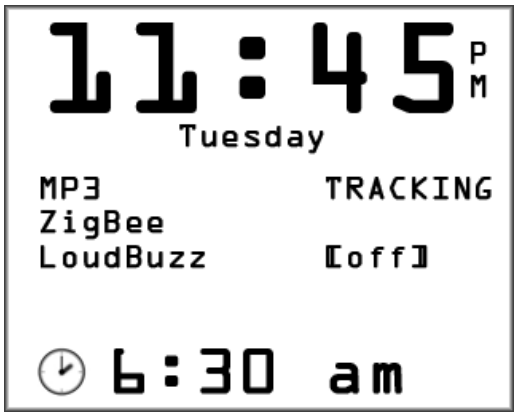


Figure 5.4.1: Example Running GUI Display

- Current Time in large display format
- Alarm Time
- List of chosen alarm actions
- Average "Time to Departure" if it has been tracked
- How many times "Time to Departure" has been tracked
- Currently playing MP3 info during **PlayMode**

Figure 5.4.2: Running Display Info List

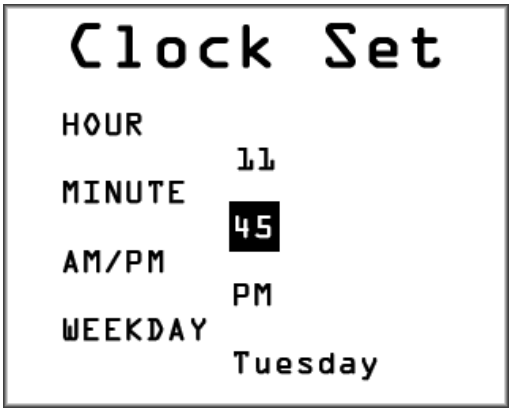


Figure 5.4.3: Example Setting GUI Display

- Clock Set
  - Hour
  - Minute
  - AM/PM
  - Weekday
- Alarm Set
  - Weekday/Weekend, 7-day, Everyday
  - Hour
  - Minute
  - AM/PM

    - Duration
- Alarm Actions
    - FM tuner
        - Order of action
        - If first action select before or at alarm time
        - if before alarm time select how early before
        - Duration
    - Play MP3s
        - Order of action
        - If first action select before or at alarm time
        - if before alarm time select how early before
        - Duration
    - Tone alarm
        - Order of action
        - If first action select before or at alarm time
        - if before alarm time select how early before
        - Duration
    - Keychain tone on Departure Time
        - Order of action
        - If first action select before or at alarm time
        - if before alarm time select how early before
        - Duration
    - Zigbee settings
        - Order of action
        - If first action select before or at alarm time
        - if before alarm time select how early before
        - Duration
        - Options available
- Enter RUN Mode

Figure 5.4.4: Tree Mapping of Setting Branch Menu

5.5 Complete User Interface

Here the interactions between PUI and GUI, and the constraints involved, are laid out in detail in figures 5.5.1 and 5.5.2.

- Hold CENTER/ENTER for 3 minutes to enter SET mode during alarm time, 3 seconds during non-alarm time
- Hold DOWN for 5 seconds to cause Zigbee keychain to beep, hold again for 3 seconds to silence
- Hold UP for 5 seconds during non-alarm time to enter PLAY for MP3 or FM
- LEFT/RIGHT tune or skip songs in PLAY mode
- PLAY mode is overridden during ALARM time

Figure 5.5.1: Running Mode Controls

- Pressing UP or DOWN will move the menu selection up and down

respectively.

- Pressing RIGHT will enter the selected setting/option/menu or do nothing if the user is already inside a value setting branch.
- Pressing LEFT will move back up out of a branch, do nothing if the user is at the top menu level, or exit a value setting branch without saving.
- Pressing CENTER/ENTER will enter a branch or save and exit a value setting branch.

Figure 5.5.2: Set Mode Controls

6.0 Sensor System

6.0.1 Motivation

The Get Up Stay Up alarm clock system requires a method of "knowing" when someone is in bed. So in an abstract way the sensor system receives the question of "is there a human occupant in the bed?" and transmits a boolean answer back to the system for it to decide what to do next. This can be accomplished in a variety of ways, including weight sensing, distance, and infrared. The system ideally will be able to tell that only one person has left the bed if there are multiple occupants, preventing conflicts if multiple users have different waking times. At the absolute bare minimum the system should not give false positives for small pets, backpacks, purses, luggage, etc. The sensor system should also be secure in that it cannot be "fooled" in any easy way that an occupant has left the bed. The sensor system should also avoid being fragile if possible, given that sleeping habits vary widely among people and sudden movements or sleep walking should not result in damage to the system or present the potential for personal injury to the user. Lastly the sensor system shall be as uncomplicated as possible so as to be conducive towards ease of installation and reconfiguration by the end user.

6.0.2 Assumptions

In the design and implementation of the sensor system the following assumptions are made. The user has a bed with at least one side against a wall, the bed consists of a mattress on either a box spring or flat bed frame, and the bed is within arm's reach of the GuSu alarm device. The mattress total weight including possible memory foam layer and sheets and pillows is less than 100 pounds. Data was collected on the beds of the four team members and roommates to create a definite picture of the average college student’s sleeping setup. This data consisted of bed height, bed length, bed width, and the distance from the edge of the bed facing the subject’s current alarm to the alarm itself. The data gathered is shown in the table 6.0.2.1, with the last row containing averaged values for reference.

Surveyed	Bed Height	Bed Length	Bed Width	Distance to current alarm
P B	61"	80"	43"	58"
J R	32"	80"	75"	19"
A L	25"	80"	52"	14"
M O	24"	78"	64"	14"
J S	38”	78”	54”	7”
B N	27”	80”	66”	6”
D D	20”	80”	53”	10”
Average	32”	79”	58”	18”

Table 6.0.2.1: Table showing collected data on bed sizes and alarm distances

6.1 Research

6.1.1 Weight Sensing

For sensing the weight of a bed occupant, options include load cells, force sensors, and FlexiForce. Load cells usually work by deformation to change electrical signals and are commonly used in structural sensing [3]. Force sensors are thin plates used for measuring surface pressure and displacement [3]. FlexiForce sensors are very thin circular force sensitive devices that produce lower resistance as the force applied increases [36].

6.1.1.1 Load Cells

Load cells could be placed beneath a mattress and can be purchased in sensitivity ranges beyond 5000 lb loads. This wide range of sensitivity makes measuring exactly how much weight has been added or removed from the sleeping area very feasible. Three viable form factors of load cells are pancake, load button, and donut types. Most load cells are cylindrical with a center pressure sensitive apparatus that changes the signal output as force is applied. Sensor thicknesses range from 0.25 inches to almost 2 inches, thus depending on the type of mattress, a solid piece of material with the same width and length as the mattress would need to be placed over the sensors and the mattress on top in order to get a proper reading of the total weight on the mattress. While the pancake and load button sensors might shift about, donut form sensors could have a small rod mounted to the bottom of this board that keeps the donut sensors in place as shown in figure 6.1.1.1.1.

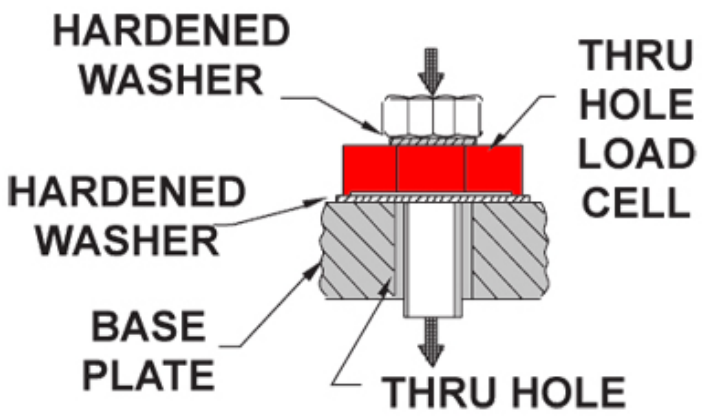


Figure 6.1.1.1.1 Schematic of a mounted Donut Load Cell
Reprinted with permission from Futek Advanced Sensor Technology Inc.

However, assuming that the user has space to spare for the sensors and the slat is hazardous, and adds unwanted complexity to installation of the GuSu system for the user. In addition it would require study on the effects adding such a layer between the mattress and bed frame or box spring would have on user comfort. Given that sensors available from FUTEK are designed for industrial OEM applications and use CC standard connection methods they would be difficult to integrate with the GuSu system.

6.1.1.2 Force Sensors

Also available from FUTEK are force sensors, available in weight sensitivity ranges from 1 ounce up to 40 pounds. These sensors are less than 0.1 of an inch thick (ranging from 0.025 - 0.05 in) making them exempt from most of the complications involved with load cells, and can be directly wired as a variable resistor. However, their sensitivity range presents a similar but new problem. Given that most bedding weighs near 40 pounds if not more the ability to measure changes in load is unattainable, and this is assuming any force sensors placed aren't already reading maximum load before any occupants have entered the bed. If the user did in fact own a memory foam layer or their particular mattress came with a detachable memory foam layer, force sensors could be used, but still would not be able to inform the GuSu system of exactly how much weight had been added or removed from the bed. Given that our environmental assumption does not include a memory foam layer, and that luggage and small pets could result in a false positive, force sensors are therefore not feasible components for this application. Figure 6.1.1.2.1 displays an example force sensor in order to demonstrate how thin most of available devices are [39].

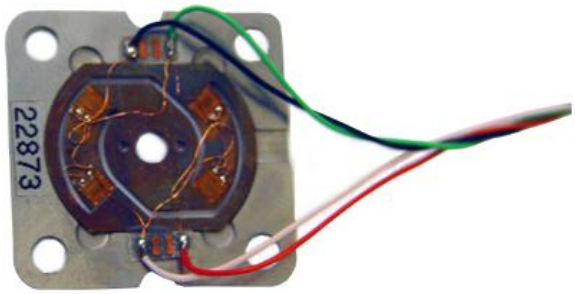


Figure 6.1.1.2.1 Example Force Sensor
Reprinted with permission from Futek Advanced Sensor Technology Inc.

6.1.1.3 FlexiForce

FlexiForce sensors, shown in figure 6.1.1.3.1, can be ordered for custom weight ranges, and come standard up to 100 lb sensitivity [37]. A convenient property of FlexiForce sensors is their sensitivity range can be tuned based on drive voltage and the feedback resistor connected in the circuit. This allows a maximum of 1000 pounds to be measured, once again making detecting different users and preventing false positives feasible. These devices also come in a thickness of approximately 0.005", and as the name implies are very flexible, reducing conditions for installation to a matter of positioning. FlexiForce sensors are also rated to handle forces up to 10,000 psi, removing any worry of damage throughout normal use.

Figure 6.1.1.3.1 FlexiForce IMAGE
Reprinted with permission from Tekscan Inc.

The arrangement and number of sensors needed depends on the sensitivity range of the FlexiForce strip. With the assumption that the user's total bed weight above the sensors is under 100 pounds, and assuming FlexiForce strips

rated for 100 pound maximum force, simply detecting in several locations a maximum load would be sufficient for proper sensing of an occupant. However, this would make it near impossible to distinguish between two occupants, and how many have left the bed. If custom FlexiForce strips are ordered the system could be programmed and calibrated to not only detect that a user's weight has been added or removed from the bed, but also could be programmed by the user for the approximate weight of any possible occupant if their individual weights differ significantly, and in turn when each needs to be awoken. This design is safe from mistakes involving most luggage and pets, and could avoid problems with couples or shared beds.

6.1.1.4 Conclusion

On the down side in order to use weight sensors the user would be responsible for the placement and calibration of the sensors. In order to silence the alarm the user could remove the sensors from under the bed when the alarm sounds and go back to sleep, defeating the key design element of the user having no easy way to silence the alarm, with the exception of leaving the bed entirely. Another problem is the added cost and the complications of if the user has a waterbed, or some non-standard bed frame, such as a bed over desk fixture, bunk bed, air mattress, etc. While most of these possibilities are not included in the assumptions made for the user environment, they are factors that limit the distribution of the GuSu system. Below table 6.1.1.4.1 compares sample products of the various pressure and weight sensing devices mentioned.

Type	Thickness (in.)	Load Range Available	Pricing ($)	Custom Order Available	Supplier
Load Cell	0.25 - 2.0	0-1000000 lb	500.00-2000.00+	No	[38]
Force Sensor	<0.1	0-40 lb	90.00-300.00	No	[38]
FlexiForce	<0.1	0-100 lb	25.00-117.00	Yes: max 1000lb	[36][37]

Table 6.1.1.4.1: Comparison of basic force/weight sensors

6.1.2 Distance Sensing

Another method of detecting an occupant in the sleeping area is to detect a change in range between some chosen point and the top of the bed. One or more sensors could be mounted over the bed in order to detect the distance to the highest point of the bed. With a twin size mattress one sensor should suffice, however double or king size beds are wide enough that an occupant could be missed. The most common methods of range detection include infrared and ultrasonic based sensor systems for finding the range to an object. Both IR and ultrasonic would be hard to use with multiple sensors positioned so close

together. Therefore to use either the system would be constrained to using only one sensor, unless one of each kind was used. However, this would still not solve the problem of ensuring the entire bedding area is checked for a human sized displacement, and with so few detectors the system could be fooled by pets and personal items. Given a user's tendency to move during sleep, the lack of full coverage of the sleeping area, and the complications with implementation of distance sensing this method of user detection is unsuitable for this application. Table 6.1.2.1 shows two distance measuring sensors available from Futurlec and Parallax.

Name	**Specifications**	**Supplier**	**Price ($)**
Sharp® Distance Measuring Sensor	Range: 100-550cm Output: 4.5-5.5V Supply: 0.5-3.0V L:60mm W: 37mm H: 20mm	http://www.futurlec.com	19.90
PING))) Ultrasonic Sensor	2cm-300cm Output: Variable width pulse Input: 4.5-6V L: 46mm W: 22mm H: 16mm	http://www.parallax.com	29.99

Table 6.1.2.1: Table comparing available distance sensors

6.1.3 Infrared

Infrared sensing has often been applied to motion and human body detection in the form of security systems and door operation. Implementation of infrared sensing falls under two main categories, emissive and passive. Emissive involves a unit radiating infrared over an area and detecting the reflections, usually in an effort to record or view a dark environment without being forced to light up an area in a way that a human eye can detect. Passive infrared sensor systems merely react to infrared radiation that passes over the sensitive elements and produces an electrical reading. In this case the system only needs to "know" if a user is present, and given that humans emit large amounts of infrared black body radiation, an emissive infrared unit is unnecessary.

For focusing or widening the effective angle of an infrared sensor, lenses can be made or attached to gather or deflect incoming radiation. Since the objective of the GuSu alarm system is merely forcing the user to rise from bed and not expel them from the sleeping area completely, a lens will most likely be needed depending on the distance and positioning of the sensor in relation to the bed. Experimentation with different distances, positions, and lenses will be needed during the second session of senior design to decide on an optimum setup, however examples of implemented infrared sensors and component specifications allow for a set of educated guesses as a starting point.

Passive infrared sensors present a good middle ground as far as meeting base requirements and optimum performance requirements. An infrared sensor or array can easily inform the GuSu system that the bed is occupied by a human and will not produce false positives due to luggage and other personal items being set on the bed. However, small pets and occupants besides the user will produce a positive signal. It is possible with adjustments to the analog to digital conversion to adjust the sensitivity of the sensor, that is setting a certain threshold for it within the converter, but most likely anybody giving off heat will activate the alarm. Perhaps this is not a bad thing as it may optionally be used to deter pets from entering the bed. An infrared sensor is also hard to "trick" short of lifting a plate of glass above the user or a small piece blocking the sensor's view. Furthermore with the sensor mounted to the wall or ceiling this won't be an easy task, and yanking on the wires connecting the sensor would damage the wall, ceiling, or the GuSu unit itself.

In the interest of user preference, Fresnel lenses could be included with focusing lenses giving the user a choice between being forced to rise from bed and being forced to leave the room entirely to silence the alarm. For this application our "default" setup will be made assuming the use only wishes to be awoken and made to leave their sleeping area. A small stand could be constructed or a camera tripod included that allows the user to position the infrared sensor on any flat surface and aim it at their bed if they prefer this over drywall mounting and have such a position available.

6.1.3.1 Infrared Lenses

Most lenses used with infrared such as Fresnel lenses are made to gather and focus so as to allow a small sensor to cover a wide area, especially in respect to security monitoring. In this case the sensor is meant to cover a specific area no larger than the size of a bed and not too small that it would allow a sleeping or waking occupant to be undetected on the bed. Glass housing would both eliminate infrared radiation from sources outside of the area of interest and make the sensor less noticeable, however using such would make production expensive and the sensor unit fragile and possibly dangerous, violating basic requirements of the system. Infrared sensors, components, lenses, and pre-made circuits for infrared motion sensing are available from many companies and in differing configurations. For quick reference during final design and build table 6.1.3.1.1 was created in case parts need to be changed.

Part No.	Description	Specifications	Price ($)	Supplier
PIR_D203B	Infrared Radial Sensor	Supply: 3-15V 120º vertical 110º horizontal	1.90	[31]
PIR_D203S	Directional Infrared Radial Sensor	Supply: 3-15V 125º vertical 42º horizontal	1.90	[31]
FRESNEL	Fresnel Lens	Used with above sensors	0.35	[31]
PIR_MODULE	PIR Sensor	Supply: 5-20V	6.90	[31]

	Module	Delay: 5s-18m Motion Detection TTL output		
PIR_MODULE_B	Compact PIR Sensor Module	Supply: 5-20V Delay: 5s-18m Motion Detection TTL output	7.40	[31]
CS9803	Infrared Induction Control IC	Supply: 4.5-5.5V CDS input Noise immunity	0.90	[31]
SEN-08630	PIR Motion Sensor	Supply: 5-12V Delay: 1-2s 0.1" pitch female connector	9.95	[39]
DEV-08645	Motion Detection Module	Low power Status LEDs on board JTAG connector Supports: Olimex, TI SBW	23.95	[39]
55-28027	Parallax PIR sensor module	Supply: 3.3-5V Parallax connectivity	9.95	[40]
AMN13111	Panasonic Electric Works Infrared Sensor	Supply: 3-6V Digital Output Settings: (standard, slight motion, spot)	33.44	[41]
AMN2111	Panasonic Electric Works Infrared Sensor	Supply: 4.5-5.5V Analog/Digital Output Settings: (standard, slight motion, spot)	12.62	[41]

Table 6.1.3.1.1: Table comparing available PIR sensors and pre-made modules

6.2 Implementation

Based on the related research, assumptions made with respect to the project, and time constraints, a passive infrared sensor with a focusing lens wired to the GuSu unit is the best candidate for implementing a sensor system that satisfies our requirements. Implementation of weight sensors coupled with PIR would be the optimum system for the reasons stated in the research section, and using both of these systems together eliminates most of their respective cons. If scheduling and progress allow during the second session of Senior Design an attempt will be made towards implementing weight sensors into the sensor system of the GuSu unit.

Given the gathered data in table 6.1 on bed heights, current alarm distance, and ceiling to floor distances, a fifteen foot cable or wire connecting the infrared sensor to the GuSu unit should be ample length to give the user a reasonable

amount of freedom in sensor placement, be it on a wall, ceiling mount, bed frame, or sensor stand. These wires will be connected directly to the in and out leads on the infrared sensor, one wire carrying the supply voltage while the other brings the output to an onboard amplified inside the GuSu unit itself. The passive infrared sensor mounted inside the focusing lens will be the directional sensor from futurlec.com (model no. PIR_D203S). With experimentation this may change, or it may be found that a focusing lens is unneeded, however as infrared reflects off walls and other surfaces it is likely a person may be detected when not in bed. This is all dependent on a room's size and layout, coupled with the distance between the bed and the sensor.

For the focusing lens itself a material must be chosen that is lightweight, easy to manipulate, and blocks infrared radiation. The GuSu unit's outer housing is planned to be made of wood, which meets all of these criteria, so purchasing lighter weight wood for the construction of the focusing lens keeps overall production simple and additional tools are not required. In addition wood is easily colored and re-colored, allowing the user to either stylize or camouflage the lens. As for the actual design of the focusing lens it should resemble lenses or fixtures that are used to focus a light source, since the sensor only needs to "illuminate" a small area, or in this case, only receive light waves from that focused area. One thing common among light focusing fixtures is their tendency to have the light source placed deep within a long, often cylindrical, housing [43]. While making a cylindrical wooden housing would be time consuming, a polygonal shape should suffice, such as octagonal or perhaps even more sides, as the more sides there are the more round the overall inner and outer surface becomes. Experimentation will be required to find the best depth and side number for the lens. Although a good starting point is eight sides with a depth from front to back of four inches. The diagrams in figure 6.2.1 illustrate the lens housing, sensor mounting, and wiring between the GuSu unit and the PIR sensor.

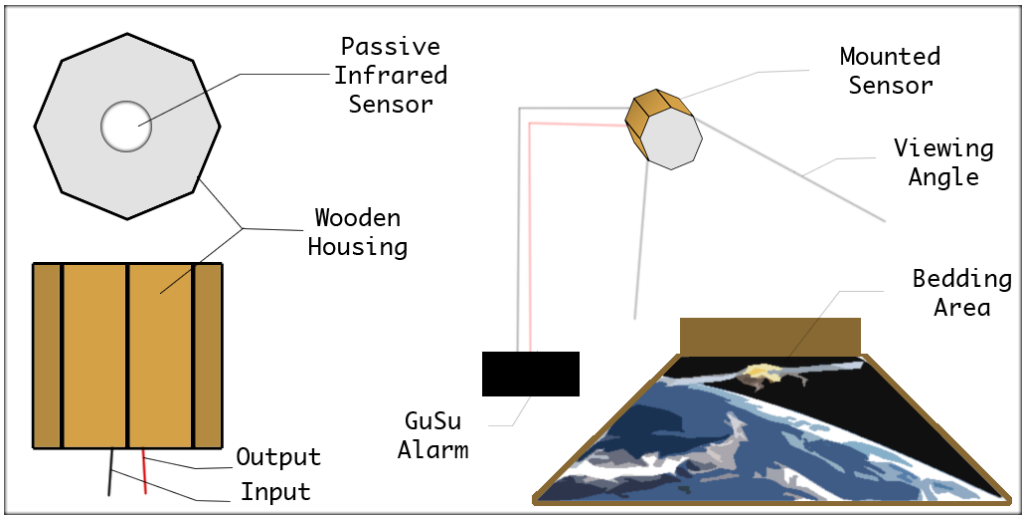


Figure 6.2.1: PIR Sensor Mounted Schematic

6.3: Sensor System Testing

The sensor system must be tested in two stages, on its own and as an input to the system. In the first case an area should be marked on either a wall or floor approximating a bed and a person should enter and exit that area. This should be done in a small room if possible to ensure that the sensor only returns a high signal when a subject is within the bed area, and not just the room. Once this has been tested any adjustments needed with respect to positioning and lens housing should be made. Next measurements of the sensor's output voltage should be taken while experimenting further with positioning, different persons, if moving to the edge of the belt still results in a high signal, and if moving towards or close to the bed also results in a high signal. To avoid false positives the GuSu's internal software should ignore quick changes.

7.0 Radio Frequency Transmission

7.0.1 Design and Requirements

The project requires secure, low power, a range of 100 meters, and two-way wireless communications. Security is a major requirement in order to prevent false triggers which could potentially lead to fires if the coffee machine should be turned on without any water in the machine. The team researched many different topologies and specifications but three wireless specifications met the minimum requirements of the project. There are many other protocols and available for use but the team felt the three listed provided the met the standards required in the alarm clock. ZigBee and Bluetooth are both IEEE ratified specifications that are readily available and Z-Wave is proprietary but is currently used extensively for home automation applications. All three topologies form a wireless personal area network (Table 7.0.1.1).

By using a standard transmitter/receiver the need to design and test radio transmission for the transceivers is eliminated as the radios already comply with FCC. Table 1 contrasts the three different topologies based on our needs.

	Security	Power	Topology	Range
Bluetooth (802.15.1)	SAFER+ block Cipher	100mW (class 1)	Star, point-to-point	~100m
ZigBee (802.15.4)	128 bit AES	1mW	Star, mesh, point-to-point	~100m
Z-Wave (proprietary)	None (required at application level)	1mW	Star, mesh, point-to-point	~30m

Table 7.0.1.1: Wireless Technologies Comparison

7.0.2 Research

7.0.2.1 Bluetooth

Bluetooth is designed as a wireless replacement for wired RS-232 communication. It is a low power communication protocol which maintains three classes (Table 7.0.2.1.1) for radio transmission. Class 1 devices met our design requirements for secure wireless communication but failed based on the power requirements for the device. The data is transmitted using FHSS (frequency hopping spread spectrum) chopping the data and transmitting it on up to 79 different frequencies. The data is transmitted over the ISM (scientific and medical) band running at 2.4 GHz short range frequency band.

Class	Maximum Permitted Power mW (dBm)	Range (Approximate)
Class 1	100 mW (20 dBm)	~100 meters
Class 2	2.5 mW (4 dBm)	~10 meters
Class 3	1 mW (0 dBm)	~1 meter

Table 7.0.2.1.: Bluetooth Classes

Bluetooth supports a number of protocols including but not limited to dial up networking profile (DUN), Personal Area Networking profile (PAN), and the service port profile (SPP). The profiles of interest to the group include the PAN profile and the SPP profile. The PAN profile describes how two or more Bluetooth enabled devices can form an ad-hoc network and how the same mechanism can be used to access a remote network through a network access point. Using this protocol the alarm clock would function as the server and the coffee machine as the end device. The Bluetooth protocol supports up to seven clients at a time. This would allow a small expansion to other devices connected to the alarm clock such as a light/lamp switch or using received signal strength indication (RSSI) a Bluetooth transceiver could be used to detect when a device such as a key chain has left the network range. This could be used to develop an algorithm to determine optimal wake up time to allow the end user to sleep until it was necessary to get out of bed (based on average times of departures). The other profile which could be potentially used is the SPP profile. This profile creates a one-to-one interface between the server and client using virtual serial ports. This would be the optimal communication protocol for our devices as the microcontrollers supports this profile and the Bluetooth devices would emulate the serial transmission between devices. The drawback to Bluetooth devices is the expense in the development and the limited availability of development kits. Sparkfun, an online retailer, provides a Bluetooth development kit. One in particular meets the SPP specifications. The Roving Networks Bluetooth module supports the SPP profile and is available in a DIP. The device does require an

RS-232 TTL converter to communicate with the microcontroller. The device is expensive costing $59.95 and the project would require a minimum of two devices for the current implementation.

7.0.2.2 Z-Wave

Z-Wave is very similar to the ZigBee specification in that it allows mesh networking between devices and was designed for home automation and sensor networks. It evolved from the 802.15.4 consortium and is now adopted by more than 200 companies [45]. Zensys, the parent company of Z-Wave, uses the same PHY and MAC layers as ZigBee but implements a proprietary Application/Framework layer. Z-Wave operates in the sub- gigahertz range and is used for low data rate transmission. The low data rate is used for on/off commands for devices such as light switches and remote controls. Metadata can also be transmitted. Since the operating frequency at the 900 MHz frequency range is not prone to the commonly used 2.4 GHz ISM frequency range employed in devices such as Bluetooth, Wi-Fi, and cordless phones. Microwaves also operate in the 2.4 GHz frequency range and can interfere with devices that use that spectrum. If the alarm clock were to be distributed internationally the 900 MHz range would need to be modified as this is not a common open frequency available in other countries as is the 2.4 GHz range. Z-Wave is designed specifically for home automation and security. This makes the protocol an optimal choice for the alarm clock project. The coffee machine, the light switch, and the alarm clock could all be monitored and maintained and controlled with a more robust PC application. The clock could be set over the wireless connection; the coffee machine could be controlled over the internet or with the clock. There are already a large number of end devices available made with Z-Wave technology and the market is well established. There is a single problem with implementing the alarm clock system with Z-Wave. The high development costs. The software development kits alone cost $3500 which does not include any hardware. A company called Control Think offers both a hardware and software development kit for only $100, but development would depend on a company with an uncertain future. Because of the high development costs for software and hardware, Z-Wave was eliminated as an option for our product.

7.0.2.3 ZigBee

ZigBee is a standardized protocol for wireless personal area networking or WPAN developed for sensor and automation applications. There are three different types of ZigBee devices: the ZigBee network coordinator, the ZigBee router, and the ZigBee end device. The coordinator links act as the master controller and links router and end devices. They form a tree network which can be seen in figure 7.0.2.3.1. The mesh uses a self healing star topology (figure 7.0.2.3.2) that allows the short range multi-hopping communication between nodes.

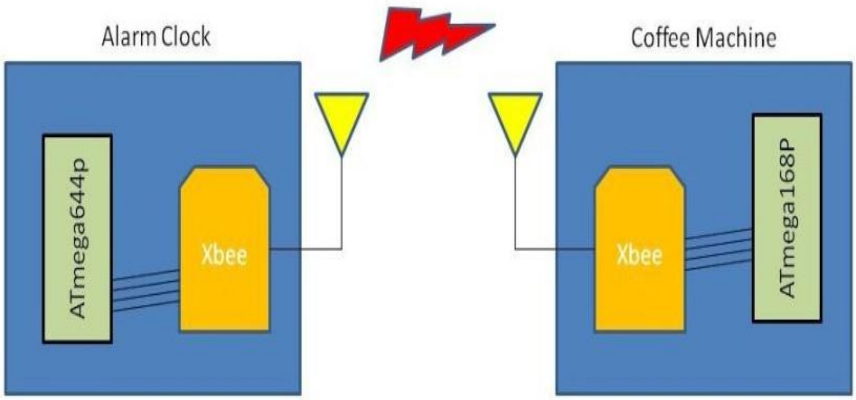


Figure 7.0.2.3.1: ZigBee Radio Transmission

Each ZigBee network can support 65,000 nodes with a single coordinator. The ZigBee specification lists the data transmission rate to be 250 Kbps but with transmission overhead the available bandwidth is 94 Kbps which is plenty for our device

.

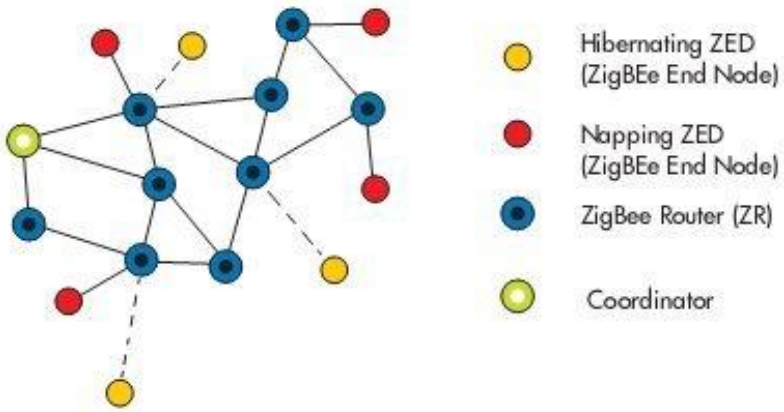


Figure 7.0.2.3.2: Zigbee network node type
Reprinted pending permission from Ember.

7.0.2.3.1 ZigBee Protocol

ZigBee builds upon the IEEE 802.15.4 standard. They implement four main layers (network layer, application layer, ZigBee device objects (ZDO's), and manufacturer specific application objects) to the MAC and PHY layers.

The ZDO's are the essential core to the ZigBee network. The ZDO layer is accountable for maintaining device roles, security, device discovery, and network joining requests. Beneath the ZDO layer is the network layer.

All routing is managed within the network layer. This layer supports star, tree, and mesh network topologies and will act as three different devices. The first is

the ZigBee coordinator (ZC). The ZC is the first required device on the network and will be implemented within the alarm clock. It initiates the network formation by selecting the channel, personal area network ID, and the stack profile. Once the network is formed it will also function as a router for other devices within the mesh. It can also act as a bind controller which will prevent other devices from joining the network without acknowledgment from the ZC. The ZigBee Router (ZR) will be the ZigBee component in the coffee machine. It is an optional component in the ZigBee network which acts as a fully functioning device and a network router to extend the range of the network. The router manages its network table for devices within range and allows neighbor routing for those devices. If a ZigBee end device (ZED) is a child of the router it will hold information for the ZED until it wakes up.

The ZED unit is also an optional device which is optimized for very low power operation. It relies on its parent node to let it sleep. It can be associated with either the ZC or the ZR but will not allow associations with other child nodes. There are multiple providers of development kits and devices using ZigBee radios. In our research the devices were narrowed down to three different devices based on availability, price, customer support, and ease of development. Initially it was intended that the product was to be ZigBee compliant using the Home Automation Public profile but found that development costs were too high for this and chose to use just the IEEE 802.15.4 specifications.

7.0.2.4.1 CC2430

Texas Instruments provides the largest number of ZigBee compliant devices and development kits. The evaluation helped to narrow the selection down to the CC2480 as it is the chip included with the eZ430-RF2480. The development included an MSP430 coprocessor chip CC2480 radio. This was our initial choice until software development costs were investigated. Texas Instruments provides their “Z-stack API” library for product development but no IDE for programming. Texas Instruments in partnership with IAR provide and IDE which integrates with Texas instruments API but even with student discounts was $800.00. As the design was to be low cost this immediately eliminated any product from Texas instruments.

7.0.2.4.2 XBee

Our second choice was the XBee radio developed by Digi International Incorporated. The chip provides UART communication using 802.15.4. The XBee-PRO ZB (figure 7.0.2.4.2.1)was the chosen module as it is ZigBee compliant if the group chooses to implement ZigBee compliance later on in our design and met all of our requirements for radio transmission.

Figure 7.0.2.4.2.1: XBee Radios
Reprinted pending permission from Digi International.

For the coffee machine, the microcontroller could use the provided digital I/O pins but opted for UART communication between the modules and the microcontrollers for potential upgrades to the system.

Pin #	Name	Direction	Description
1	VCC	-	Power Supply
2	DOUT	Output	UART Data out
3	DIN/Config	Input	UART Data in
4	DO8	Output	Digital Output 8
5	Reset	Input	Module Reset
6	PWM0/RSSI	Output	PWM Output 0 / RX Signal Strenght Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR/Sleep_RQ/DI8	Input	Pin sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4/DIO4	Either	Analog Input 4 or Digital I/O 4
12	CTS/DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	On/Sleep	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate/AD5/DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	RTS/AD6/DIO6	Either	Request to Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3/DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2/DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1/DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0/DIO0	Either	Analog Input 0 or Digial I/O 0

Table 7.0.2.4.2.2: XBee Pin Assignments
Reprinted pending permission from Digi International.

The XBee ZB modules have an indoor range of 133'. Adding more devices to the network can increase the network range but using a single module for the network should be adequate for most households. Transmit and peak current are each at 40mA while the power-down current is less than 1 µA. The module handles all the data communication and is self healing. If a node in the system

fails, the module will automatically seek another node to rejoin the network. The modules support serial communication speeds of 1200-230400 bps (and non-standard baud rates). They also provide small buffers to collect received serial and RF data. Since the data packets will be small, overflow should not be a problem with the system. Overflow can occur if the module receives a continuous stream of RF data and the transmit buffer is also being overloaded. Another problem can occur if the network link is not established and the buffer is continually loaded with data to transmit. In order to prevent the errors, the system will not send any serial data until network links are established and as previously mentioned data transmissions will be short.

7.0.3 Implementation

The XBee modules as indicated above will only be transmitting serial data over the ZigBee end devices. Each XBee module will be connected to a microcontroller. The devices will be pre-programmed at set channels and data rate. In order to configure the devices for serial commands they will be connected to a USB to serial converter board which is configured to work with the XBee module. Sparkfun distributes a PCB with an integrated USB to serial converter and provides the pin headers for the non standard XBee modules. Each device will be configured and with two separate XBee explorer USB devices and tested as mentioned in the test plan section and configured as follows. Maxstream, the manufacturers of the XBee module, provides software named X-CTU which is used to configure the modules. XBee configuration is started with the command '+++'. Once the XBee transmits 'ok', configuration data can be sent to the module. Once each configuration level is sent, the configuration parameter will be checked by sending the same command without the parameter. If configured correctly, the entered data should be echoed back to the console.

The default settings allow other XBee modules to communicate with each other to prevent interference from other modules or in the event that another user is using the same configuration parameters, the settings will be customized. There are multiple parameters for each of the modules to talk to each other. Each of the parameters needs to be configured and saved to ensure proper communication. Each module must be on the same network by setting the ID parameter (see configuration below for details on setting the parameters). The modules must also be on the same channel by setting the CH parameter. The baud rate of each device must also be set at the same rate. Finally, each module's destination address (destination high (DH) and destination low (DL)) must be configured. These parameters determine which modules on the particular and network and channel will receive the data it transmits. The destination high and destination low parameters can be configured in a few ways:

- If a module's **DH** is 0 and its **DL** is less than 0xFFFF (i.e. 16 bits), data transmitted by that module will be received by any module whose 16-bit address **MY** parameter equals **DL**.

- If **DH** is 0 and **DL** equals 0xFFFF, the module's transmissions will be received by all modules.
- If **DH** is non-zero or **DL** is greater than 0xFFFF, the transmission will only be received by the module whose serial number equals the transmitting module's destination address (i.e. whose **SH** equals the transmitting module's **DH** and whose **SL** equals its **DL**).

7.0.4 Configuration

The available configuration options available for using the XBee modules are listed in table 7.0.4.1. There are other commands supported, but the XBee module will only be used as a virtual serial port between the microcontrollers, only the options listed below will be used.

Command	*Description*	*Valid Values*	*Default Value*
ID	The network ID of the XBee module.	0 - 0xFFFF	3332
CH	The channel of the XBee module.	0x0B - 0x1A	0X0C
SH and SL	The serial number of the XBee module (SH gives the high 32 bits, SL the low 32 bits). Read-only.	0 - 0xFFFFFFFF (for both SH and SL)	different for each module
MY	The 16-bit address of the module.	0 - 0xFFFF	0
DH and DL	The destination address for wireless communication (DH is the high 32 bits, DL the low 32).	0 - 0xFFFFFFFF (for both DH and DL)	0 (for both DH and DL)
BD	The baud rate used for serial communication with the Arduino board or computer.	0 (1200 bps) 1 (2400 bps) 2 (4800 bps) 3 (9600 bps) 4 (19200 bps) 5 (38400 bps) 6 (57600 bps) 7 (115200 bps)	3 (9600 baud)

Table 7.0.4.1: Common Serial Configuration Parameters
Reprinted pending permission Digi International.

Serial communication will be transmitted at 9600 baud using the configuration command ATBD 3. The channel for each module will be for the alarm clock and the coffee machine 0x11. Any additional XBee devices will be configured with same channel ID as this is required for communication over the network. The network ID is configured by sending the command ATID XXXX, where XXXX is the ID number in the range of 0 to 0xFFFF. Initially this will be set at a static value of 0xAAAA. If the alarm clock system goes to market, the alarm clock master ZigBee module can be set to scan for any transmitting channels. If one is found, this channel would be eliminated from the selection as the alarm clock would be functioning as the master controller and another channel is chosen.

Again if the product went to manufacturing, the coffee machine would be set to scan all the network channels and look for the alarm clock and send commands with expected return values. As noted in the table 7.0.4.1, DH and DL settings are specific to the operation of the modules. The alarm clock, acting as the master controller, will have the DH address as 0 and DL as 0xFFFF. The coffee machines DL and DH addresses will also have the same initial address as the alarm clock. Using the same parameters eliminates any link communication problems. Of course this will need to change if the device does go to market. As a note, once all the configuration parameters have been set, the command ATWR is sent to the XBee module and the configuration parameters are written to NVRAM on the module. Once this is done, the XBee modules will be connected to each of the respective microcontrollers. As there will be different microcontrollers, each separate implementation is listed in the following sections.

The alarm clock will be using an ATmega644p which provides two separate UART pins. The LCD module will be connected to one set and an XBee module will be connected to the second set. On the microcontroller, pin 16 (RX1) will go to pin 3 (DIN/CONFIG) on the XBee module. Pin 17 will be connected to pin 2 (DOUT) on the XBee module. These two pins handle the data transmission. For the power, pin one is VCC and connected to a 3.3V source and pin 10 is connected to the common ground of the system. All interface and link information will be displayed on the OLED module. The coffee machine will be hooked up in the same manner as the alarm clock system. In addition to the connections from the microcontroller, LED's will be hooked up to the XBee module to indicate communication and link status. The microcontroller and the XBee module will both be running at 3.3 V. Pin 7 will be hooked to power and pin 1 will hooked to power. The common ground will be connected to pin 10 on the XBee adaptor and pin 8 on the microcontroller. Transmit (pin 3) on the microcontroller will be connected to receive (pin 3) on the XBee module. Receive on the microcontroller is pin 2 and it will be connected to pin 2 on the XBee module. There will be two LED's on the coffee machine. The first will indicate if the coffee machine is in the on state (i.e. coffee is brewing or already brewed). This LED will be connected to a digital output pin on the microcontroller. The second LED will be connected to pin 6 on the XBee module to indicate data transmission. This will give an indication to the user that the system has successfully linked up. The details of the coffee system are in the 7.2.

7.0.5 Test Plan

The microcontroller will only be using the UART mode of the XBEE modules. For testing the function and communication, the group will use two XBee USB to serial adaptors. The devices will be configured as listed in the implementation section above. Using a windows DOS terminal, the microcontroller will send a series of commands and verify receipt of the command. This will be tested from both devices.

7.1 Remote Detection Unit

7.1.1 Technical Objectives and Specifications

In order to optimize the wake up time for the user, the group decided to implement a remote detection system. The device would be a wireless transmitter/receiver that the alarm clock would ping at set intervals to detect when it was out of range. Initial suggestions were using a key chain system that would run on battery power. Upon determining the power requirements of the XBee module, this would not be a viable choice as the battery would only last approximately two to four hours depending on the frequency of transmission. The next option would be to place the XBee module within a car. It would be powered off the car battery which eliminates the power system dilemmas.

7.1.2 Goals

The alarm clock will be able to determine if e the user is out of range of the alarm clock system. The alarm clock will be able optimize the wake up time to ensure the user has left the premises by the time specified by the user. The system will be able to detect the differences between device out of range and device failure. The alarm clock software system will be able to determine whether the car transceiver has left range or if it has simply failed.

7.1.3 Input/output

The remote (car) transmitter will simply be powered by the car battery. It will be a standalone device which will be programmed to send short packets to the alarm clock. The XBee modules support RSSI to indicate when the range changes. If the RSSI is decreasing the alarm clock will request transmission frequently to ensure the device is leaving by verifying a progressively decreasing RSSI and making sure it is not just interference or a dropped communication.

7.1.4 Power

Car batteries typically run between 12V and 16V. The XBee module require 3 – 3.4 V for the Pro version. To bring the power down to specifications a LM3703 will be used. It is a linear voltage regulator available from multiple manufacturers and can drop the voltage to 3V. A simple PCB with the voltage regulator and the XBee module would be all that is required.

7.1.5 Software

There is no software requirement for the key chain system. All Zigbee modules support node discovery (ND) operations. The node discovery command can be used to discover all modules that have joined a network. Issuing the ND command sends a broadcast node discovery command throughout the network. All devices on the network that receive the command will send a response that includes the device's addressing information, node identifier string (see NI

command), and other relevant information. This command is useful for generating a list of all module addresses in a network. The node identifiers are unique to each module and can distinguish between the coffee machine system and the remote sensor unit. When a device receives the node discovery command, it waits a random time period before sending its own response. The maximum time delay can be set on each individual module using the ND command for the sender with the NT command. The ND originator includes its NT setting in the transmission to provide a delay window for all devices in the network. Large networks may need to increase NT to improve network discovery reliability. The default NT value is 0x3C (6 seconds). With only three devices on the network, the default settings will be kept at 6 seconds. There is also an option to have the remote sensor act as a beacon transmitter. In beacon mode, the transmitter could transmit at a random 30 second interval. If the received signal strength was significantly decreased between pings, the transmitter could decrease the ping time by half with each successively decreasing ping time. The base transceiver (alarm clock) could measure the RSSI with each ping and determine if the receiver were actually leaving the network area.

7.1.6 Research

The research for wireless communication modules was limited to Zigbee devices as Zigbee is already limited in the system. This limits production costs and development costs. The XBee module supports received signal strength indication so the alarm clock will be able to support the distance/range detection. There are a number of other Zigbee modules (including those mentioned in the wireless communication section) that support RSSI. No further investigation was conducted on Zigbee module based on previous investigations and design costs and time.

7.1.7 Implementation

As with the other XBee modules the all device configuration will occur prior to installing it. The device only requires power to function. It will only support node discovery. There will be a single LED connected to the remote signal strength indicator pin (pin 6) to ensure remote connection to the Gusu alarm clock system. It will be lighted when a connection is present. The remote detection unit will be mounted in a sealed plastic enclosure and will have an external 12V cigarette power adaptor to be plugged into the automobile. Being that there are very few components in the system, it will be mountable in a small enclosure. The soap box enclosure (figure 7.1.7.1) available from Sparkfun.com will be large enough to contain all of the devices.

Figure 7.1.7.1: Soap Box Case for remote detection unit
Reprinted with permission from Sparkfun Electronics.

Since the remote sensor system will not include a microprocessor all of the processing will be performed at the master controller unit (alarm clock). The alarm clock will send out the network discovery command every thirty seconds. If the remote module is determined to be in the process of leaving the network by measuring a decreasing RSSI signal strength, the alarm clock system will decrease the ping time to 6 seconds. It will not be set lower than six seconds as this is the timeout period for the network discovery command. Once three successive pings have been acknowledged with three successively decreasing signal strengths have been received, the system will consider that the node has left the network. In order to minimize the network traffic the ping time will be set to 30 minute intervals during non alarm time interval. The alarm time interval will be defined as the time period between 30 minutes before alarm wake up time to an hour and a half after alarm trigger. The circuit is quite simple (figure 7.1.7.2) and will be built with a perfboard as it only requires four components.

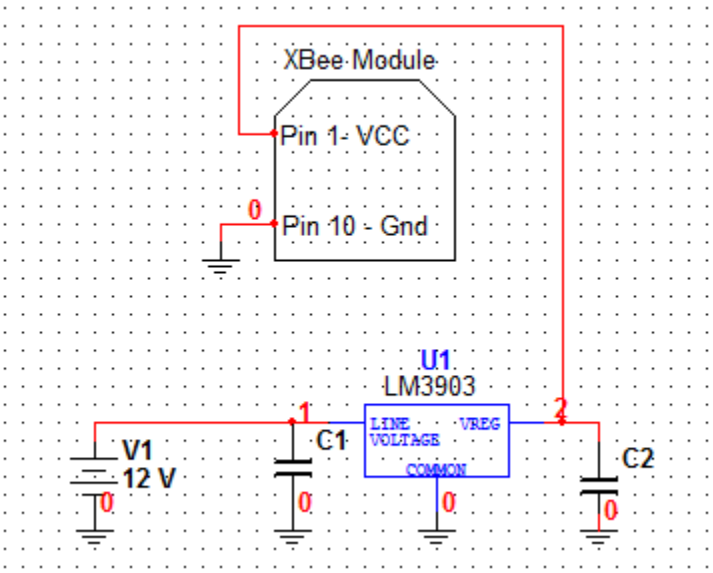


Figure 7.1.7.2: Circuit diagram for remote Detection Unit

7.1.8 Test Plan

In order to test the remote sensing unit, two Zigbee modules will be used. The first will be considered as the master and the second as an end device. Each will be programmed with the same network ID and the location ID. The master will be controlled using the programming software X-CTU provided by Digi International. Using the software, the microcontroller can monitor the received signal strength of the end device. Range testing will be performed using different materials in each of the designer's homes. The materials include concrete block, wood, and metal. The group will also test the range in different weather conditions to test the effects rain/water could have on the sensitivity. Measurements will be taken to determine the optimal ping or beacon times to test for the sensor dropping out of range at the speed of 25 mph.

7.2 Coffee Machine

7.2.1 Technical Objectives and Specifications

7.2.1.1 Goals

An off the shelf simple coffee machine with only on/off functions will be purchased and modified for the project. The coffee machine will be externally connected over a wireless ZigBee mesh network. All communications with the device will be over secure network communications which will prevent the coffee machine from turning on at random periods. The coffee machine will be controlled with microcontroller and communicate with the alarm clock via UART to the XBee module. It will have a timer to automatically shut off the coffee pot after 30 minutes from time of turn on and this will also be communicated back to the alarm clock over ZigBee. There will be a single button on the coffee machine to locally toggle the machine on and off for unscheduled activations. There will also be a single LED to indicate power on and ZigBee link status.

7.2.1.2 Input/output

With only a single button and a single LED only two digital IO pins will be required. For the XBee Module the alarm clock microcontroller will need a single UART port on the microcontroller. To activate the brewing heater for brewing the coffee the coffee system will need a digital relay to enable the heating element. Since standard coffee makers have built in sensors to manage the heating, this will be left to the pre-existing hardware. The system will only be controlling the on/off state of the coffee machine.

7.2.1.3 Power

Coffee machines run off standard 120 VAC. The microcontroller and XBee device require VDC. During the research DC powered coffee makers were found. These coffee machines are primarily used in boats, campers, and for

frequent travelers. The RoadPro 12 volt DC powered coffee machine appears to be one of the few available systems. It of course plugs into a 12V DC outlet which is non standard in homes. The power requirements of the coffee machine were noted to be 130 watts and 11 amps. The tremendous power draw would require a large ac/dc transformer which is prohibitively expensive costing around ~ $150.00. This will necessitate an ac/dc converter within the machine. There are three considered options available to convert the power. Full wave rectifiers with DC voltage regulators, transformers with voltage regulators, and switched mode power supplies are the researched options. Batteries could also be used to power the DC components but even with every DC electronic system in lower power state, the system would last only approximately 3 days before requiring new batteries. The problem with modifying an off the shelf system is room for power conversion circuits. The smallest switching mode power supply found during the research was approximately 2'' x 2''. With the microcontroller and XBee module already occupying the limited space the group felt that the module power supply could be installed externally if necessary. As the coffee machine has not been obtained yet, this will be investigated further during development. The group consensus is to currently supply the DC power via a second power source. An AC/DC transformer can be housed external to the system and heat problems will also be eliminated. The same power design implemented with the alarm clock system could be used to power the microcontroller at the full 5V and a step down voltage regulator could be inserted to power the XBee module at the require 3.3V. The other option would be to power both the XBee module and the microcontroller at 3.3V as previously mentioned with a single 3.3V DC wall wart. If the power problems prove difficult to implement, there are multiple manufactures which produce ZigBee compatible relay switches which can be controlled by the alarm clock.

7.2.1.4 Software

The software will only need to handle controlling simple communication with the XBee module and enabling and disabling three separate I/O Pins. As discussed in the Clock System, by using similar microcontrollers the group will be able to save time and money in development and cost. By using a similar architecture the group will not need to learn new programs for development and buy separate development boards.

7.2.1.5 Physical Package

The microcontroller for the alarm clock includes a number of pins and features that are not necessary for the clock but ATMEL makes a number of microcontrollers that support the same 8-bit RISC architecture with smaller footprints.

7.2.2 Research

Since the group will be restricting the microcontroller to an Atmel device the following device topologies from Atmel will be contrasted. The first device is the ATiny 2313.

7.2.2.1 ATtiny 2313

The Atmel ATtiny 2313 met the requirements all of the requirements set forth in the previous section. It is capable of handling UART communication with the XBee module and has multiple digital I/O pins for controlling the relay. It can also be powered at the same voltage as the XBee module. The ATtiny 2313 is an 8 bit RISC processor with 2 Kbytes of in system programmable flash memory. It can operate from 0-20 MHz on a voltage range of 1.8V – 5.5V. There are 20 pins total with programmable I/O pins for connecting to the relay for the heater element and the LED lights for user updates. There is a single UART port for communication with the XBee module. The only negative factor is the programming of the microcontroller. Atmel provides an IDE to develop for the board in either C or Assembly languages. The group is leaning towards using the open source Arduino boot loader so this eliminates this microcontroller as it is not compatible with Arduino.

7.2.2.2 ATmega168

As this microcontroller is similar to the ATmega644, only the finite details/differences will be listed in this section. The other Atmel chip researched is the ATmega168 which is compatible with the Sanguino boot loader. The packages available are a 28 pin DIP, 32 pin VQFN, and 32 pin TQFP packages. The DIP package is preferred for its ease in prototyping. It is also an 8 bit RISC microprocessor with 16Kbytes of programmable flash memory and 512 bytes of EEPROM. It supports 131 instructions and 32 x 8 general purpose registers. Like the ATtiny2313 it will operate from 0-20 MHz at a voltage range of 1.8V – 5.5V. A single UART is available for communication with the XBee ZigBee radio modem. There are 23 other I/O pins available for analog or digital signals. A single external interrupt and internal interrupts are also supported. The microprocessor also supports five software selectable power modes (Table 7.2.2.2.1). The power-down mode saves the register contents to NVRAM and freezes the internal oscillator until an interrupt or hardware reset on pin 1. Power save mode keeps the asynchronous timer running while shutting down all other functions on the device. ADC Noise reduction mode allows the microprocessor to shut down all I/O modules except the analog to digital converter to minimize switching noise while sample analog signals. In standby mode the oscillator continues to run which permits fast startups and low power consumption. The idle mode stops the CPU but leaves the USART, SPI, interrupts, SRAM, and the timers running.

	Active Clock Domains					Oscillators		Wake-up Sources						
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other/O
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X(2)	X(3)	X	X(2)	X	X	X	
Power-down								X(3)	X				X	
Power-save					X		X(2)	X(3)	X	X			X	
Standby(1)						X		X(3)	X				X	

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

Table 7.2.2.2.1: Power Modes for ATmega microcontrollers
Reprinted pending permission Atmel.

7.2.3 Implementation

The ATmega will be running at 3.3V because the XBee module also requires the same voltage. The microcontroller will be communicating with the XBee module using UART pins 2 and 3, RX and TX respectively. It will also be connected to an LED with a single Digital output pin. The led will indicate to the user that the coffee machine is in the ON state. A single relay will also be connected to a digital output pin of the microcontroller. As most motors and relays are inductive, they shouldn't be connected directly to a microcontroller because they may require more current than a microcontroller can safely supply. The requirements for the relay are 3.3V and can tolerate 10 amps. According to the U.S. Department of Energy, the average coffee maker consumes 800 -1200 watts. This equates to 6.7 to 10 Amps. As the coffee machine has not been selected at this time, consideration of power consumption to ensure that it is less than 1100 watts will be taken. Once purchased, current consumption will also be measured to ensure manufacturers specifications.

In order to protect the microcontroller from the inductive loads a transistor will be used as a switch to prevent damage to the microcontroller during relay switching. A diode is also needed to protect the microprocessor from back EMF current and is placed in parallel with the relay and the power supply (figure 7.2.3.1).

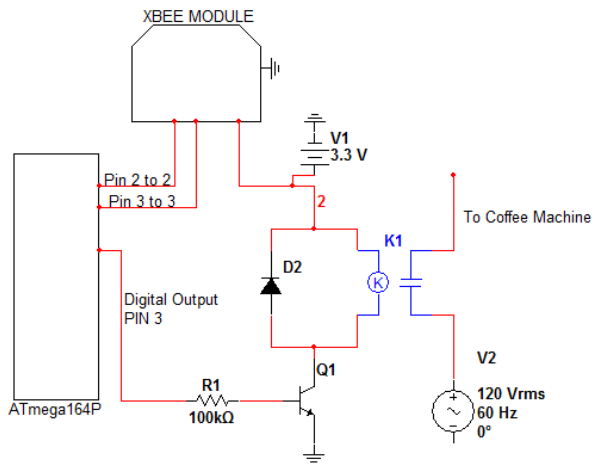


Figure 7.2.3.1: Circuit Diagram for Coffee Machine System

As the only DC power supply available in the coffee machine system is 3.3V a 3V relay must be used. The only available relay found during research was the Omron G6C-2114P-US-DC3. It can handle a resistive load at 10 amps and 240 VAC. The software for the system is very simple. Once power is applied to the coffee machine system, the XBee module joins the network using previously defined connection settings which is described in the Radio communication section. The system then goes into low power idle mode waiting for user input from the local power toggle or a transmission from the alarm clock. The activity diagram below (figure 7.2.3.2) outlines the functions of the system.

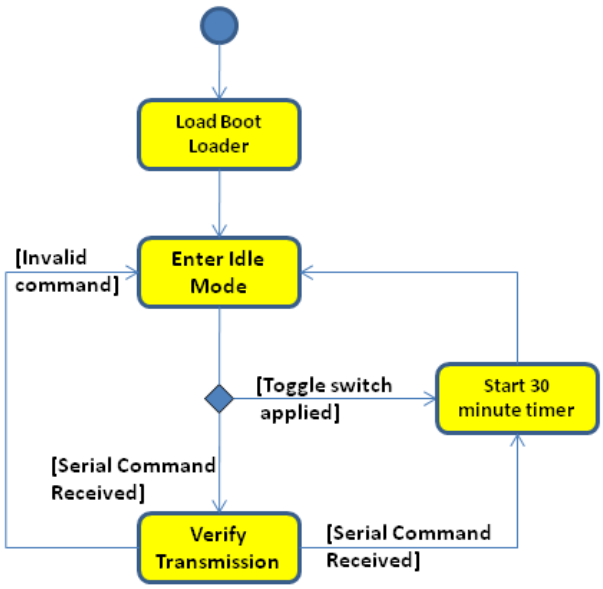


Figure 7.2.3.2: Activity diagram for coffee machine system

As you can see, the coffee system is quite simple. Verifying transmission is handled by echoing back the message to the alarm clock system and receiving the same instruction previously received. This assists in preventing false triggers and thus potential fires. The coffee machine has a built in temperature regulator so this does not need to be monitored by the system. A single counter is started once the button is pressed or the correct serial command is received from the alarm clock. Until the timer has expired, the system will ignore all commands from the user and remain in the ON state.

7.2.4 Test Plan

The microcontroller will be booted and serial communication will be tested with a terminal server and a PC. An LED will be hooked up and tested by toggling it on and off. Since the system is quite simple, the code will be developed and completely tested on the system using a terminal server. Timeout verification will be verified by sending an on command and confirming that system has returned to idle mode and confirming that the LED is off.

8.0 Clock Module

8.1 Goals and Objectives

The alarm clock system will need to maintain accurate time with little drift. It will also require the ability to maintain a minimum of two alarms to be stored in the system. These can be stored externally or in NVRAM on the microcontroller.

8.2 Research

Two different clock modules were investigated for the system. Ideally the atomic clock was to be implemented as it would limit the need for user interface design and contribute to the ease of use. Since there is limited availability atomic clock modules real time clock modules were also investigated.

8.2.1 Atomic Clock

WWVB broadcasts a radio time signal from the National Institute and Technology (NIST) in Fort Collins, Colorado. The signal transmits both time and date 22 hours a day. Using a radio receiver the clock can automatically set itself. The radio clock module would, in the event of power failure/loss, transmit the correct date and time to the microcontroller without requiring the user to set the time. This would simplify the programming of the microcontroller by limiting the implementation of the user interface. This the most desirable choice but the availability of clock modules is very limited. In fact, only one such module could be found that was available for purchase. This was the CME6005 made by C-Max Time Solutions. It is an IC that requires external antennae design and fabrication and research has shown that the device is fairly unreliable. The cost of the radio clock module and complexity of including it in our design is also

much greater than a real time clock module eliminating this option from project consideration.

8.2.2 Real Time Clock

The alarm clock will contain a real time clock (RTC) module. This will allow the clock to maintain time, date, and alarm settings in the event of power failure. The RTC will also provide a number of integrated features which can be used in our system. The countdown timer can be used to begin counting when the alarm goes off and gradually increase the volume of the alarm until the user gets out of bed. There are many interfaces available but in order to keep the complexity and cost of the system low, the group opted for a real time clock with an SPI interface. As with the other components, a dual inline package will be chosen.

Two devices meet our requirements. The first is the DS-1305N (figure 8.2.2.1) available from Dallas Semiconductor and the Intersex CDP68HC68T1. Each module meets all of our requirements. The DS-1305 provides a few more features such as dual alarms and costs half as much as the Intersil module. Clearly the choice will be the least expensive module with more features.

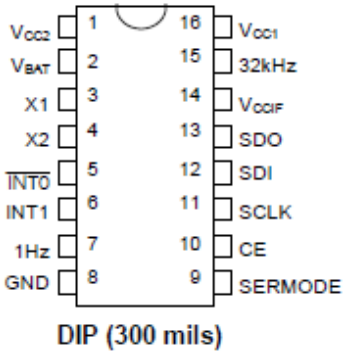


Figure 8.2.2.1: DS-1305 Pin Diagram
Reprinted pending permission from Maxim.

The DS-1305 is a 16 pin DIP with a clock/calendar which provides seconds, minutes, hours, day, date, month, and year information. All of this data is accessible over the SPI interface. The module also provides 96 bytes of NVRAM for storing data such as alarm settings. The alarms can be set to output a single or different interrupt in the event of an alarm condition and can be operated under either of the available alarm conditions. Dual–power systems are supported allowing for a backup battery in the event of power failure and the V_{BAT} pin outputs a programmable trickle charge permitting the use of rechargeable batteries (figure 8.2.2.2). The operating voltage is from 2.0V to 5.5V which falls within the range of other modules within the system.

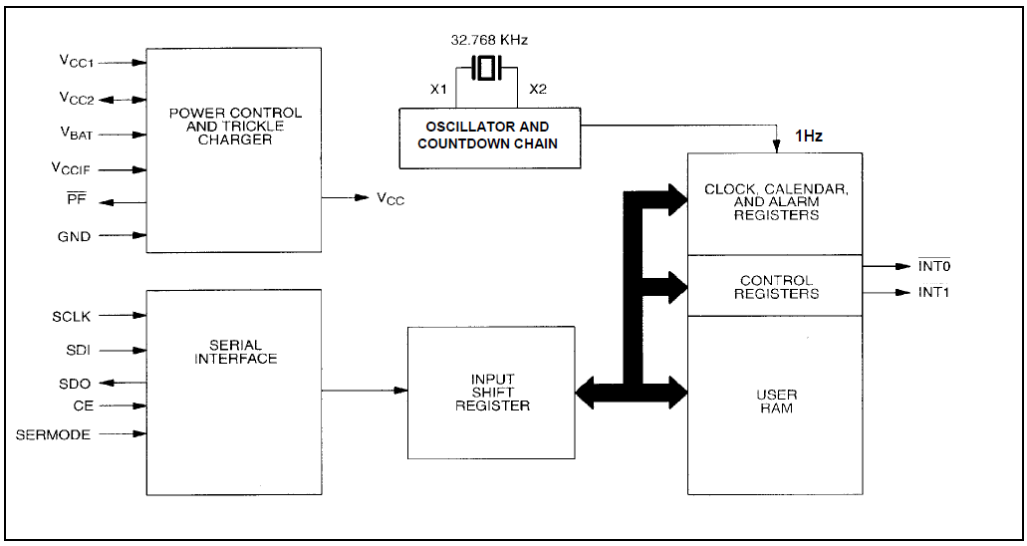


Figure 8.2.2.2: Block Diagram for DS-1305
Reprinted pending permission from Maxim.

The clock, calendar, and alarm times can be obtained by reading the appropriate register bytes. The time settings are initialized by writing to the specific register bytes. The contents of each register are stored in BCD format. The day values are stored as sequential user defined numbers. For example, if the 1 equals Sunday then Monday would be the next sequential value of 2. If any of the register values are improperly set, the operation of the clock is undefined. The DS1305 can operate in either 12-hour or 24-hour modes. In order to configure the hour mode bit 6 is toggle high or low. When high, 12-hour mode is selected. The rest of the register settings are listed in table 8.2.2.1.

HEX ADDRESS		Bit 7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	RANGE
READ	WRITE									
00H	80H	0	10 SEC			SEC				00–59
01H	81H	0	10 MIN			MIN				00–59
02H	82H	0	12	P/A	10-HR	HOURS				01–12 + P/A
			24	10						00–23
03H	83H	0	0	0	0	0	DAY			01–07
04H	84H	0	0	10-DATE		DATE				1–31
05H	85H	0	0	10-MONTH		MONTH				01–12
06H	86H	10-YEAR				YEAR				00–99
07H	87H	M	10-SEC ALARM 0			SEC ALARM 0				00–59
08H	88H	M	10-MIN ALARM 0			MIN ALARM 0				00–59
09H	89H	M	12	P/A	10-HR	HOUR ALARM 0				01–12 + P/A
			24	10						00–23
0AH	8AH	M	0	0	0	0	DAY ALARM 0			01–07
0BH	8BH	M	10 SEC ALARM 1			SEC ALARM 1				00–59
0CH	8CH	M	10 MIN ALARM 1			MIN ALARM 1				00–59
0DH	8DH	M	12	P/A	10-HR	HOUR ALARM 1				01–12 + P/A
			24	10						00–23
0EH	8EH	M	0	0	0	0	DAY ALARM 1			01–07
—										
0FH	8FH	CONTROL REGISTER								—
10H	90H	STATUS REGISTER								—
11H	91H	TRICKLE CHARGER REGISTER								—
12–1FH	92–9FH	RESERVED								—
20–7FH	A0–FFH	96-BYTES USER RAM								—

Table 8.2.2.1: DS1305 Registers
Reprinted pending permission from Maxim.

The DS1305 also contains two times of day alarms which can drive two external interrupts. The alarm times are set by writing to registers 87h to 8Ah for alarm 0 and registers 8Bh to 8Eh for alarm 1. The alarm clock will be using two alarm clock settings for days of the week. The module supports five different settings for alarm time/day triggers by setting bit masks in each alarms respective registers. The five settings are alarm once per second, when seconds match, when minutes and seconds match, when hour's minutes and seconds match, and when day, hours, minutes, and seconds match the set time. During each clock update, the RTC clock compares both alarm 0 and alarm 1 registers with the clock registers. If a match occurs, the corresponding alarm flag bit is set to 1. If the corresponding alarm interrupt is enabled, the external interrupt is driven high. Data is transferred between the microcontroller and the RTC when the CE pin is driven high. A single byte is transferred per clock cycle of SCLK at the shift edge when selecting read or writes to/from registers. Address and data bytes are shifted MSB first. All transfers require address of the byte to indicate write/read to the RTC or to the memory location which is followed by the data. Once the register has been selected the data can be read in burst mode. Each read/write cycle causes the RTC to automatically increment each data member in the register until the device is disabled (setting CE low). One precaution is that while reading or writing in burst mode, the address pointer will wrap around after reaching 1Fh for reading and 9Fh for writing to the clock. It will also wrap around when accessing the RAM after reaching 7Fh for reading and FFh for writing.

8.3 Implementation

The DS-1305 is available in a 16 pin dual-inline-package making it easy to integrate into the PCB. The design uses the SPI interface so the microcontroller will need to have SERMODE pin (pin 9) connected to VCC_1 (pin 16) to enable the SPI interface. The alarm clock system design includes a battery backup module for the entire system so V_{BAT} (pin 2) will be used and the pins for battery charging will be left open. The module requires a 32.768 Hz crystal to be connected to X1 and X2, pins 3 and 5 respectively. Pin 8 provides a 32.768 Hz output that can be used throughout the system. If the alarm settings are to be implemented in the clock module, both interrupt pins can be used independently from one another. Pins 7 and 8 both output separate programmable interrupts for alarm signals. Each of these pins will be tied to digital inputs on the microcontroller. The rest of the pins used will be for serial communication between the clock module and the microcontroller. CE (pin 12) must be enabled for all reads and writes to the clock module. Once asserted, pins 14, 15, 16 enable all communication with the module. Pin 14 is the serial clock pin used to select which device to use in all SPI communications. This pin will be tied to pin PB7 on the ATmega644p. SCO and SCI (pin 15 and 16) will be tied to PB5 and PB6 for data transmission.

8.4 Test Plan

The clock module will maintain the critical function for the alarm clock in keeping time. All used functions of the module will need to be tested thoroughly. Since the alarm clock system will be using the SPI interface, the module will be tested by hooking it up to the ATmega644p SPI interface after the microcontroller has been tested. All pins will be wired as indicated in the implementation described above. The clock time will be set and power will be applied for 24 hours. Once the time has elapsed, the module time will be compared with a synchronized clock on a nist.time.gov (an online official time clock from the U.S. government which is accurate within 0.2 seconds.) Once time maintenance is confirmed, the backup battery functions will be tested by removing primary power for 30 seconds and re-applying power. If the device maintains correct clock time the alarm interrupts will be tested using interrupts on the microcontroller and output interrupts on the clock module.

9.0 Power Supply

The alarm clock will require power to operate, and this section is dedicated to the supply and backup storage of that power.

9.1 Block Diagram

Below (figure 9.1), is a very top-down block diagram of the power system of our alarm clock. The two main components of note in the design of this section would be the Power Supply and the Battery Backup.

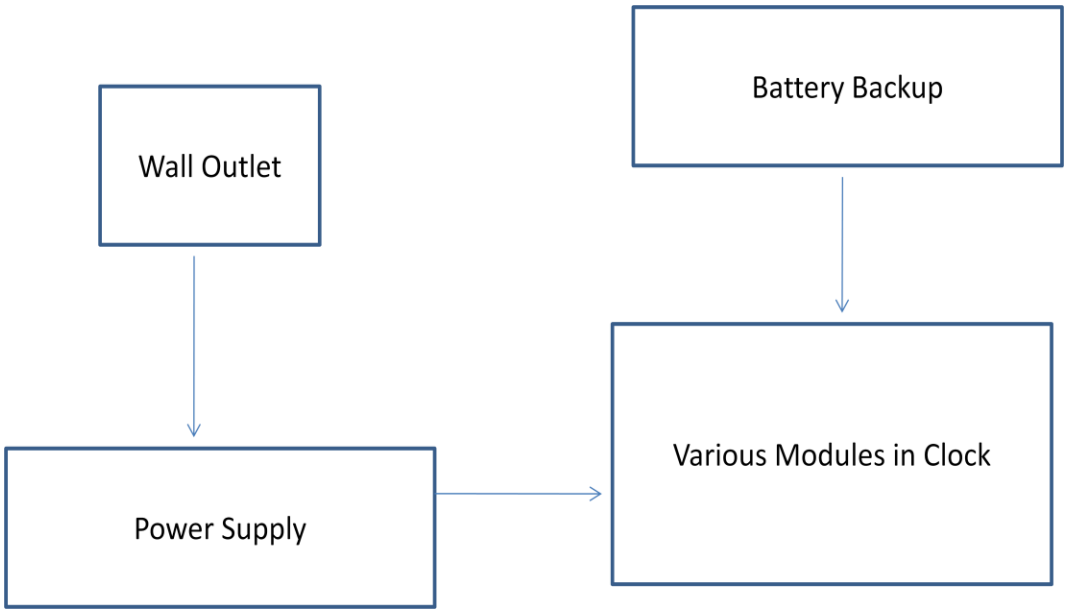


Figure 9.1: A block diagram of the "power system."

9.2 Power System

9.2.1 Power Supply

The alarm clock will be powered through power available from wall outlets. This voltage will be regulated to a DC voltage of around 3.3 to 5V, depending on the needs of the used components.

9.2.1.1 Research

Of course, everything in the electronics world needs power. In this project's case, every component from the microcontroller, the tuner, and everything in between, will need to be powered by DC current with a voltage of 3-5V. Because America (and much of the rest of the world) generates AC power for distribution, that power must be converted from AC to DC. This can be done in a variety of ways, but first it is important to understand how it works.

The first step in any power supply is to convert the 110V AC voltage to a DC voltage, preferably somewhere between 9V and 15V [53]. This is often done with something known as a 'wall wart,' or with a power brick. These are basically the same thing; the main difference is that wall warts plug into a wall and output the desired voltage directly out form there through the cord, whereas power bricks are fed a line from the wall to the device where the A to D conversion takes place. This is then fed through a second line to the device that is consuming the power. The main advantage of wall warts would be the easier design, and it only needs one cord. Another advantage is that is keeps any heat from the conversion right next to the wall outlet, and far from the device. While power bricks can be kept from the device as well, a user that does not know any better

could set the power brick directly by it or under it. The main advantage for power bricks, however, would be that they do not take up any more space on the outlet than necessary. Oftentimes, wall warts can be quite bulky, blocking access to other plugs that could use the same outlet. The same is true of plugging them into power strips. An example of both the wall wart and the power brick is shown below in figure 9.2.1.1.1.

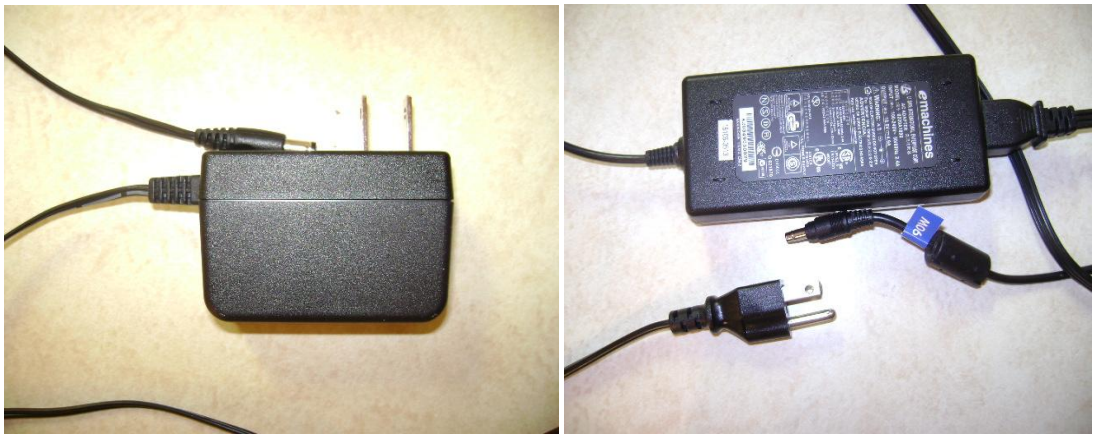


Figure 9.2.1.1.1: Both a wall wart (left) and a power brick (right). Notice how the wall wart could potentially block access to a lower outlet, depending on its physical configuration.

Wall warts or power bricks for this project could come from a variety of places. Many home appliances use them, and one could rather easily be scavenged for use in this alarm clock. Each one plainly and simply states the input and output voltage and current levels. It would be important that a suitable one is used if this is the option the group decides to take. For example, one with too high of a voltage level would likely not cause too much of an issue (see below about regulators), but one that does not output enough current could cause problems. They can also readily be found online. For example, many wall warts were able to be found for under 20 dollars on Amazon.com, with outputs ranging from .5A to 2.5A [54]. Power bricks are readily available online as well, with an example of a 12V power brick being available from store.mp3.com for $29.99 [55]. The many options will make the choice a somewhat simple one. For the GuSu alarm clock project, it really does not matter if the group uses a wall wart or a power brick. The important thing will be that it supplies a decent voltage, and enough current to supply all of the components.

The voltage that comes from this wall wart (or power brick) is a noisy voltage, somewhat around what is specified. For example, a 9V wall wart will not give exactly 9V; more likely it will rise and fall somewhere between 8.5 to 9.5 V [53]. This somewhat-DC voltage needs to be both dropped down to the needed voltage (around 3.3V to 5V), and cleaned up (so it will be a steady, flat voltage rather than one with ripples). This can easily be accomplished with the use of a regulator. A common three-pin regulator is shown below in figure 9.2.1.1.2:

Figure 9.2.1.1.2: A common three-pin regulator.
Picture reprinted with permission from Sparkfun Electronics.

A 5V regulator could take any DC voltage from roughly 7V to about 34V and output that voltage at 5V [53]. A very common part for this task is the LM7805. Referring to Figure 4.3.2.2, the first pin on the left would be connected to the input voltage from the wall wart, the middle pin to ground, and the output pin would be 5V. Now if the input to the regulator is a noisy 9V, the output would be a noisy 5V. This can be fixed using capacitors to smooth the voltage levels. Without heavily going into the physics of capacitors, they basically provide stability to the voltage level, by absorbing energy when the voltage is high and expelling energy whenever the voltage is low. As long as the voltage is not dropping out for considerable periods of time (power outage), and there are not significant spikes in the voltage levels, the capacitors will keep the voltage to a steady 5V. The higher the capacitance rating of the capacitor (in Farads), the more energy it can store, meaning the more it can release. However, the higher the capacitance, the slower the storing and releasing of energy takes place, and vice versa. So, with most regulators, it would be important to place larger capacitors in the range of 10 to 100 micro Farads on the power rail [53]. These capacitors help to hold the voltage up if the power falls for a short time (about 10 to 100 ms). An example of the previously mentioned LM7805 5V regulator is shown below in Figure 9.2.1.1.3, with the 10 uF and 100 uF capacitors on the 5V and 9V sides of the regulator.

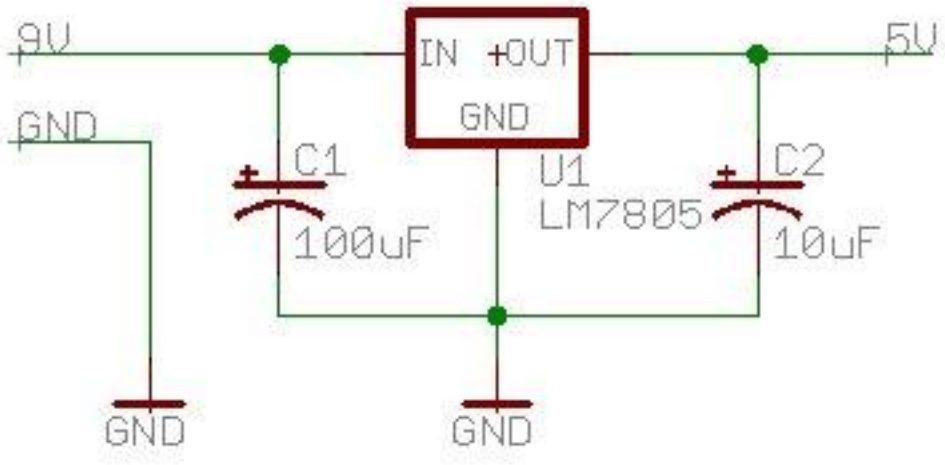


Figure 9.2.1.1.3: LM7805 regulator with capacitors.
Reprinted with permission from Sparkfun Electronics.

These voltage regulators are rather easily found online. From the Sparkfun Electronics website, a LM7805 5V regulator with a max current flow of 1.5A is available for $1.25 [53]. An LM78M05CT 5V regulator with max current flow of .5A is available from www.national.com for a mere $0.58 [56]. However, these come with minimum orders of 45, so that would not be very feasible. Many companies offer samples of these, and some requests could be put out if the group believed that it was needed. There is a wealth of other companies selling similar voltage regulators for similar prices, and it would mainly be very important that the regulator purchased for this project could support as much current flowing through it as the clock would need. However, the alarm clock should not need more than 500mA of current.

According to Sparkfun Electronics, it is also very helpful to place a diode at the 9V receiving end to ensure that the current will only flow in one direction [53]. This can be important in some instances where the connection might be made many times, or by the user. This would prevent the accidental incorrect wiring of the voltage line and ground, resulting in a reversing the polarity of the voltage, and thus the direction of the current. However, because this part of the project would not be touched by the user, the diode would not be a necessity. It will just be important for the group members to keep the voltage and ground lines straight during development.

One thing that would be very handy, however, would be a way to know whether or not the power is on. An indicator of sorts would be nice to have, so if there were ever any problems during testing (which is very possible), rather than having to check the voltages with a multimeter each and every time, it would be nice to see a straightforward visual indication. This can rather easily be done by using an light-emitting diode (LED). Without going explicitly into how LEDs work, suffice it to say that they emit a small light when a small amount of current flows

through them. In general, LEDs should never have more than 20mA of current flowing through them, as around that point they begin to fail [53]. The easiest way to ensure the reduced current in the LED is to put it in series with a resistor. The resistor would be chosen, of course, based on the voltage. So if there was a 5V regulator, and someone wanted to know how to tell if that regulator is truly outputting about 5V, an LED and a resistor could be wired in series from the 5V voltage level to ground. In this example, the resistor to be chosen would probably be about 330 Ohms. Using Ohm's Law that would create a current of about 15mA, which would be more than enough to light the LED. Now, if there were ever significantly less than 5V coming out of the regulator, the LED would be off to indicate that. Chances are the voltage out of the regulator would be nearly 5 or nearly 0. Because of that, if the light is lit up, than there is a good chance that if anything is wrong, it is not the power supply. An example of a circuit with the LED installed is shown in Figure 9.2.1.1.4:

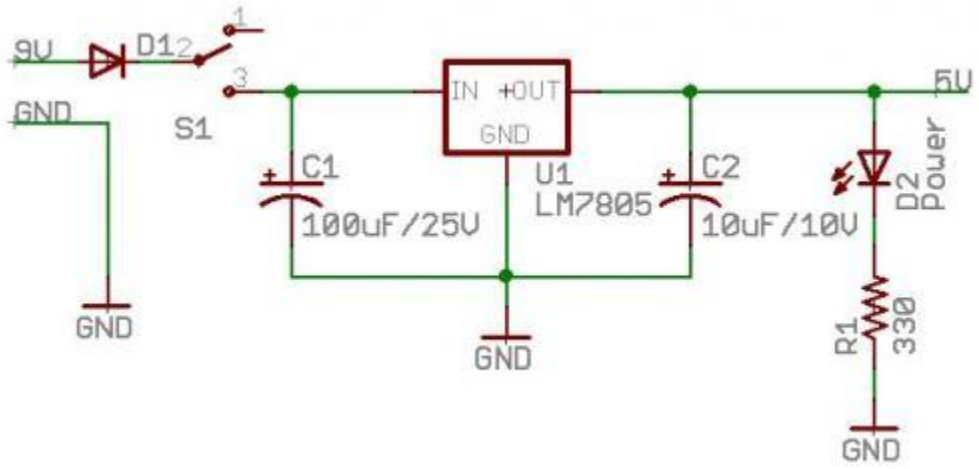


Figure 9.2.1.1.4: The LM7805 regulator with capacitors, a switch, and the LED installed. Notice that it makes no difference whether the resistor or the diode is placed first in the series. Reprinted with permission from Sparkfun Electronics.

Different colored LEDs can be found for prices that approach being free with minimal searching. A quick search of SparkFun Electronics found red and yellow LEDs for only $0.50 each [53]. There are many other suppliers, and it could be added to any other orders from most electronics websites.

The "build it yourself" method outlined above would not be too difficult. It would require some PCB (printed circuit board) space, possibly its own small board for the regulator. It could probably be integrated onto another board to save size and costs, however. The regulated voltage, if chosen to be around 5V, could then be stepped down as needed to power lower voltage elements, if any exist in this project.

Another option for the power supply would be to feed pure 110V AC voltage into a PCB that included an all-in-one Analog to Digital converter, which should

output a regulated voltage at the level needed for this project. A search of Acopian Power Supplies, a well-known power supply delivery company, shows all-in-one PCB parts with a regulated 5V output ranging from $79.00 to $97.00 [57]. These prices are reflective of an output current max of .5A (model 5E50A) and 1A (model 5E100), respectively. This max current would of course depend on the requirements of all the modules in the project. These also come with guaranteed tolerances of +- 0.1% for the .5A model and +-.15% for the 1A model. These all fit within a small roughly 1X2.5X1 inch package.

This alternative would make the design much simpler. Rather than dealing with all the parts and building the PCB for the first method, this could very easily be integrated into the main microcontroller circuit, being fed simply by just a wall plug. However, they are pretty pricy. Especially if this were to ever try to be implemented in a marketable fashion, there is no way to justify spending over 75 dollars for a power supply that could be implemented with under 25 dollars. While it will definitely be an option to consider in the design phase, depending on the complexity of the rest of the project, it is unlikely that it would be the better option.

Earlier, it was mentioned that a step-down may be needed in some cases. This is because it can oftentimes be much easier to take our regulated 5V DC voltage and step it down to the desired voltage (e.g. 3.3V) than to build a second entire power supply (wall wart, regulator combo). That much is obvious. Of course, a simple method to produce 3.3V from 5V is to put two resistors in series from the 5V line to ground, with the relative resistances being 1.7:3.3. For example, a 170 ohm resistor in series with a 330 resistor would drop 3.3V between them if they were put in series in that order. However, with this configuration, it would actually be very important to know a constant load being drawn from the stepped down voltage, as this load would be in parallel to the 330 ohm resistor. If it was anything other than much higher than 330 ohms, it would affect the actual resistance seen from the 5V line, and affect the dropped down voltage. Another more feasible option, in many circumstances, is to purchase a DC voltage step-down converter.

There are a couple different types of voltage converters, and basic ones can be implemented very easily, if there is a known input and output voltage. An example of a 5V to 3.3V step-down/step-up is shown below in Figure 9.2.1.1.5. This is the LM3350 from National Semiconductors, and is available from Digi-Key for $2.02 per unit [57]. The advantages of such a converter are the ease of installation, the size (the example below is about .2X.1 inches), and the low cost. The disadvantage is that it is not variable, so the required output voltage must be known in advance.

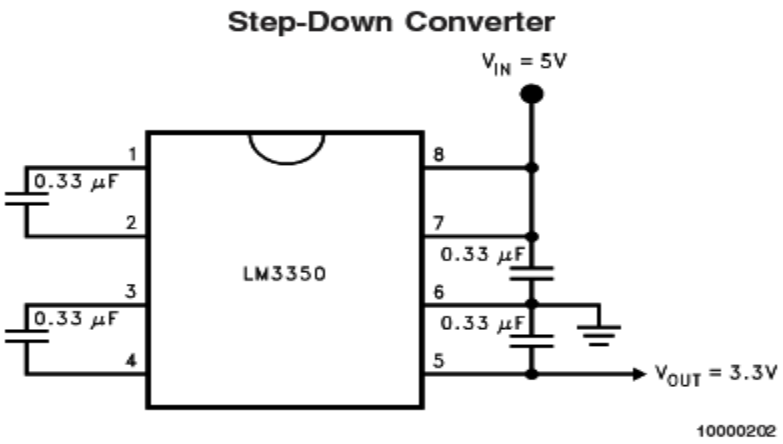


Figure 9.2.1.1.5: The LM3350 5V to 3.3V Step-down converter. Reprinted with permission from Digi-Key.

Another option is a variable output voltage converter. One of these is a 10W adjustable step down switching regulator, DE-SWADJ [58]. This is adjustable with a screw, and could easily be set to the desired output voltage while using a multimeter. This specific model costs $15.00, and shipping is $1.25 internationally. While it is slightly more expensive, it needs no external parts, and would be much easier to implement. The fact that it is adjustable makes it easy to use with changing objectives in an initial design such as this one. This way, if the project's power needs change, the step-down voltage can change with it, without rendering any previously built PCBs and designs worthless. One disadvantage is that it is larger, but still being roughly the size of a quarter in length and width, and maybe half that size tall. Not needing the external parts that the above LM3350 would need makes a big difference in the PCB design.

9.2.1.2 Implementation Strategy

For this project, it was decided to use a +12V DC wall wart to provide the power. This is convenient, mainly because of the Op-Amps that require a +12V and -12V biasing. The 5V regulator should perform just as well with the 12V incoming as it would with the 9V. The wall wart to be used will be converted from an older wireless network router wall wart that is rated at 12V DC up to 1A. The part that used to be plugged into the router will be split, and the +12V and ground lines will be soldered to the PCB to provide for the power rail and ground plane.

It is important, before going further, to know the voltage and current requirements of all of the devices to be used in the clock. Together, they cannot exceed the ability of the power supply to deliver it. Table 9.2.1.2.1 shows the separate devices, and their power requirements that are known from researching their datasheets (unknown values will be determined experimentally when the devices arrive).

Device	Voltage Req. (DC)	Current Req. (Active)
Microcontroller	2V – 5V	<10 mA
FM Tuner	4.5V – 5V	8mA
LCD Screen	4V – 6V	10-115 mA (typ. 40)
PIR Sensor	3V – 5V	<100uA
Buzzer	4V - 6V	30 mA
Mp3 Decoder	2.4V – 3.6V	<30 mA
SD Card Reader	3V	
Clock/Timer	2V - 5.5V	2 mA
ZIGBEE	2.1V – 3.6V	40 mA
Op-Amp	+12V and -12V	
Totals	2.4-3.6, 4-5, -12, 12	250 mA max

Table 9.2.1.2.1: Table of the different components in the GUSU alarm clock, and their voltage and current requirements.

As can be seen in the table above, the minimum amount of voltages required are a +12V and -12V to bias the Op-Amps, a voltage between 2.4V and 3.6V to satisfy the Mp3 decoder and Zigbee components, and a voltage between 4V and 5V to be a sufficient source for the rest of the components. The max current that could be drawn if all of the components were drawing max current at once would be about 250 mA. Of course, that is not possible. For example, if the buzzer is drawing current, then the FM Tuner, Mp3 decoder, and SIM card reader all would not be, and vice versa. Regardless, it is best to plan on a higher max current draw than is possible so that there are no problems. In that regard, it is important to make sure that all components involved in the power supply are rated to handle a current of 500 mA flowing through them. The above table can also be used to construct a simple block diagram of the power supply system, which is shown in Figure 9.2.1.2.2.

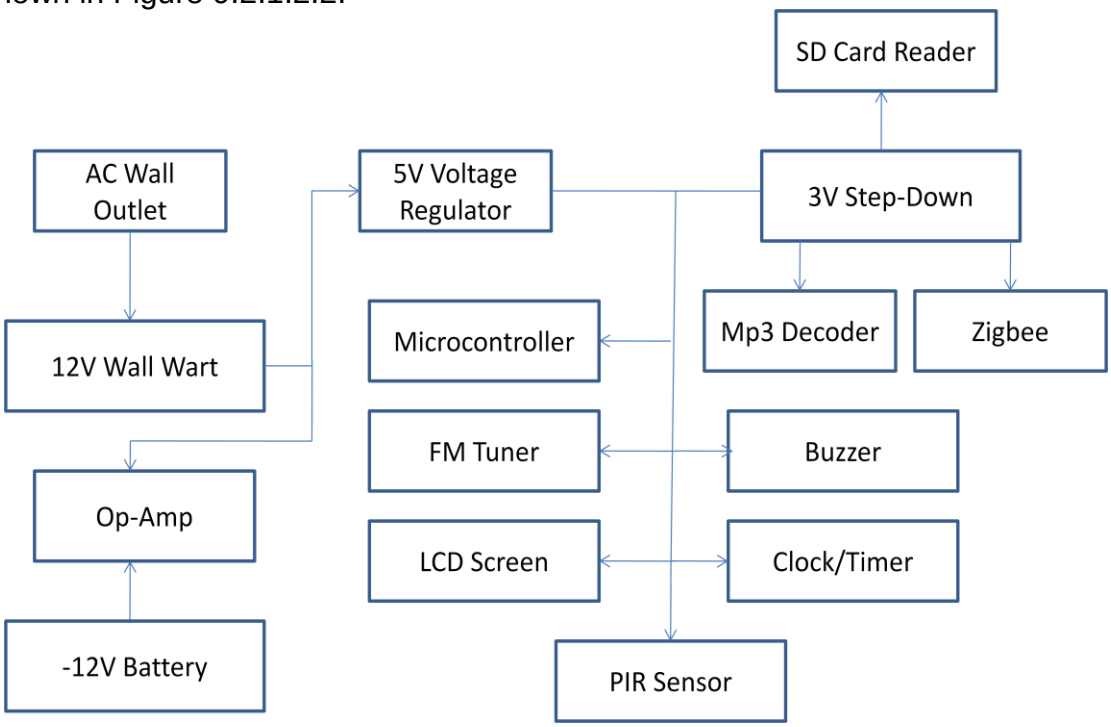


Figure 9.2.1.2.2: Block Diagram of the Power Supply System.

The +12V and -12V are taken care of through the use of the 12V wall wart and the second battery (discussed in Section 4.2.1.2: Audio Amplification). The 12V power rail will just need to be a thin trace (thin because of low current draw) over to the Op-Amp from the larger trace to the voltage regulator on the PCB. The voltage regulator to be used is the LM7805CT-ND 5V 1A regulator, manufactured by Fairchild Semiconductors, and purchased from Digi-Key for $0.45 each. This regulator takes the input 12V and outputs a steady 5V (According to specs, 4.8V to 5.2V, which should be good for all the components). This voltage regulator is implemented with the circuit below in figure 9.2.1.2.3. The two capacitors help to stabilize the DC input and output in case of short power drops (for drops in power on the order of 10 to 100 ms). The LED, as explained in the research section (9.2.1.1), is positioned with the resistor so that only 15.15 mA should flow through it. This is to prevent the excess of current from damaging the component.

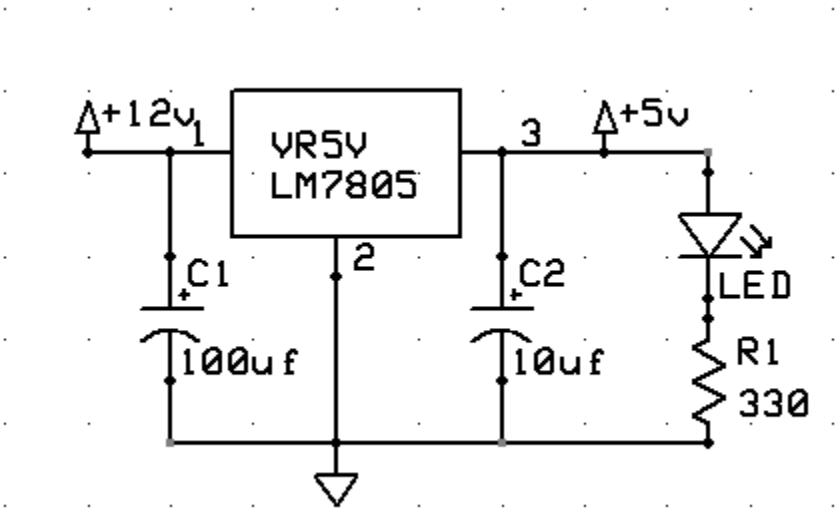


Figure 9.2.1.2.3: The 5V voltage regulator.
Schematic created using ExpressSCH.

This 5V line needs to be 'stepped down' to a value between 2.4V and 3.6V. This value will be chosen as 3V for simplicity. The part chosen to do this is the DE-SWADJ, purchased from Dimension Engineering for $15.00. This part takes any voltage from 3V to 30V and will step it down to a DC voltage between 1.25V and 14V. The voltage is adjusted through a small screw on the side of the component (a potentiometer). As the screw is twisted clockwise, the output voltage is reduced. The efficiency is rated at 92%, although with only two smaller components drawing from the 3V source, it should not matter too much. Same is true for the max current flow, which is 1A. The final schematic for this step down is found in Figure 9.2.1.2.4.

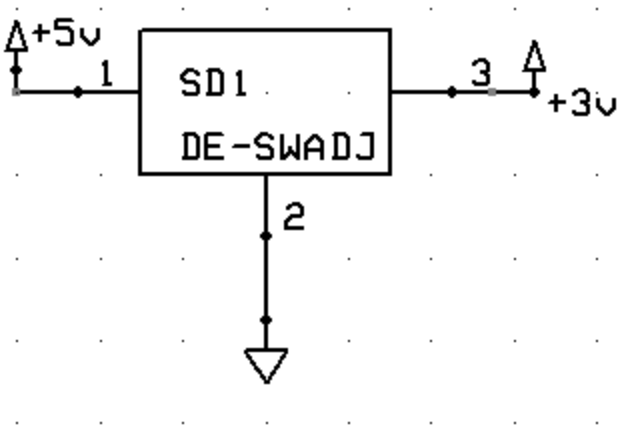


Figure 9.2.1.2.4: The voltage step-down device.
Schematic created with ExpressSCH.

PCB implementation will be discussed in Chapter 10.

9.2.1.3 Test Plan

It will be important to know that all of the different devices in the GUSU clock are able to operate off the main power supply without needing to draw current from the backup. The following test procedure should ensure this.

1. Remove backup battery supply (keeping the -12V battery, B3, installed for the biasing of the Op-Amp).
2. Turn on the clock.
3. Begin testing each of the components (the test plan for each component should be found in each components respective section).
4. Using a voltmeter, test the voltages at Vcc (or Vdd) and ground for each of the components found in Table 9.2.1.2.1 (the device power requirements list). This is especially important if any devices fail their tests, and it is not readily apparent that the problem is with the device itself.
5. If there are any problems, trace them back to the sources (i.e. the 12V input, the 5V voltage regulator and the 3V step-down). If these voltages are at their correct values, then so should all of the device voltages!
6. If there is a discrepancy between the voltage at a source and on the device, there may be a problem with the PCB traces themselves. It may be worth attempting to remove the trace and hard-wire the voltage to the device.

Assuming the power system works as it should, the battery backup test could be initiated afterwards.

9.2.2 Battery Backup

Battery backup is considered to be an integral part to the Get-Up Stay-Up alarm clock system. This is because if the user decided that he/she would like to sleep through the noise, the only thing the user would have to do is remove the power plug from the wall. That would kind of make the entire project pointless. A worthwhile battery backup would not need to last long; with the noise planned on coming out of it, less than 5 minutes should be needed to fully wake any sleeper. It is just important that the user cannot disable it easily, so the battery backup would need to be encased with screws needed to remove the batteries. The backup would also be able to keep the clock time set during short power outages. The specifications for the device are that it is able to provide uninterrupted power for a full hour when unplugged from the wall with a fresh battery, that it does not drain the battery when there is current flowing from the wall outlet, and that there is minimal lag during the switch, so the devices do not lose information.

9.2.2.1 Research

An easy way to incorporate battery power is to stick commonly available batteries prior to the 5V regulator. While this is a waste in a way, only using 5V of the batteries outputting voltage level, it is a simple way to incorporate the battery backup. The wastefulness likely would not impact the user, as the battery will almost never be used. This is not for an emergency radio that could be used in the event of hurricane-like power outages, it is meant to last long enough to wake the user if the user unplugged the clock from the wall. Six AA batteries could be placed in series to come up to a total voltage of 9V DC. This voltage could differ of course, based on the needs of the design. According to Techlib.com, the average AA battery lasts about 2000mAhrs [59]. So even if the clock used a total of 250mA, which is an extremely high estimate, the battery backup would last over 8 hours. In fact, 2 or more sets of the AA batteries could be placed in parallel to increase this estimate. But with just one set, it would last long enough for the user to completely wake up after unplugging the alarm clock as many times as necessary to break them of that habit. It would also hold the power steady during small power outages, no problem.

One important thing to consider is that the batteries should not be draining while the alarm clock is plugged in. This could be accomplished easily with one major constraint: whatever the final power supply voltage is prior to the regulator, the voltage supplied by the batteries must be less. If this is the case, a simple diode placed heading from the battery supply to the voltage from the wall wart would keep the current from exiting the battery supply (there would have to be roughly a .7V drop between the two for current to flow. In this case, if the input voltage dropped significantly (at least .7V) below the battery supply level, the battery current would kick in. It would be extremely important that the battery supply was lower than the voltage coming from the wall wart/power brick, as if it were higher, the current would flow through the diode anyways. It could be accomplished by

simply stacking as many AA 1.5V batteries as necessary without going too high. It is important that it is above about 7 V however, so that the 5V regulator has enough voltage to account for the drop it needs), so it makes sense to go to about 9V as long as the wall wart is above that level.

9.2.2.1 Implementation

The entire battery backup implementation is simple enough. It just requires 8 AA batteries in series, or 1 12V battery (A23 model, by Duracell), followed by a diode. With a wall wart with a voltage greater than or equal to 12V, the current will never drain from the batteries except when the main power supply is unplugged. This is because of the .7V drop on the diode that is necessary in order for current to flow from the battery backup. The line coming out of the diode connects to the line from the wall wart to the DC 5V regulator. A schematic of this can be seen in figure 9.2.2.1.1.

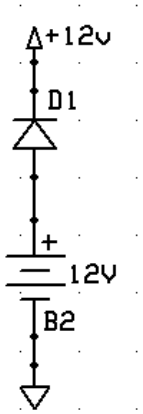


Figure 9.2.2.1.1: Schematic of the battery backup system, made with ExpressSch. The DC voltage is actually the 8 AA batteries in series (or 1 12V battery), in a small case made to hold them as compactly as possible.

The battery backup would not take any space on the PCB, as there is a premium for that space (the larger the board, the higher the cost). Instead, the battery backup will be attached to the case in some manner, with leads going from either end of the battery and being soldered into two holes in the PCB. A small casing can be constructed from wood, or other materials to hold the battery backup, and can be screwed into the inside of the casing.

9.2.2.2 Test Plan

The test plan for the battery backup is actually quite simple. All that is necessary is for the wall wart to be unplugged from the wall outlet, and to see if the clock is still working. It would be important that all parts of the device are working properly. The test plan for this group will be as follows:

1. Remove 12V wall wart from wall.
2. Ensure display is still functioning, and that no memory (such as date and time, etc.) has been lost.
3. Remove wall wart from wall during the alarm, and ensure that there is no overly significant loss in volume, that the sensors still work properly, and measure how long it can be kept functional by draining the batteries.
4. Ensure that when the main device power supply is enabled, no current is being drained by the batteries. If this is the case, it may be necessary to lower the voltage on that line, as well as the Op-Amp negative bias line (they should be the same, so that neither the positive nor the negative quantities of the audio signal would be amplified more than the other). 9V batteries could be used in this case as opposed to the 12V batteries (or 6 AA batteries as opposed to 8 AA batteries).

Assuming that the test plan worked as above, the batteries would need to be replaced, be checked for proper voltage levels, and the device plugged back in to avoid draining the newer batteries.

10.0 Software

10.0.1 Motivation

In order to bring all of the systems and functions of the GuSu alarm system together in an automated fashion, a form of logic must be built in to the system itself. An incredibly complex artificial intelligence could serve to expand functionality and introduce auto calibration and so on. But this is highly unnecessary as the requirements and design of the overall user interface and GuSu system provide ample flexibility for user preference and system performance. The following sections discuss alternatives and software choices and then go on to explicitly describe the software that will be written for the GuSu system and how it will behave.

10.1 Software Research

10.1.1 Alternatives

The GuSu system architecture could be controlled and run solely by hardware using binary logic. Binary clocks, abstracted boolean logic, transistors to flip on and off the various alarm systems and other controls could theoretically accomplish anything software would do. However, as possible as this is, it is highly implausible and would greatly increase the time for design, construction, and debug of the system. User control would become excessively complex, and as college students the team does not have access to materials and equipment for making miniature electronics, and this if the system was built to run on hardware only its size and weight would increase beyond the specifications chosen for the GuSu unit's housing. Similar discussions involved implementing

the internal clock using binary logic gates but the advantages of ordering a component that is pre-made and tested far outweighed the disadvantages of going ahead with making one from scratch. Thus all of the GuSu alarm system's functions and sub systems will be controlled through software run on the central microprocessor.

10.1.2 Programming Languages

The microcontrollers investigated to run the GuSu device interpret C, C++, and the descriptive programming language Arduino. Since C++ has all the abilities of C and more, it of course would be very advantageous to use over C. Arduino, which is C-like in syntax, has the advantage of being structured and created as a Processing/Wiring language, reducing the amount of abstraction between physical microcontroller hardware and software written to control it. While C++ is an excellent language for higher level systems and complex software packages, the GuSu alarm system in the grand scale of things is a rather "simple" system overall and would benefit more from the use of Arduino than C++. To further this point the two languages are compared below in the following table.

Feature	**C++**	**Arduino**
Compiler/Developer Environment Available	Visual Studio, Xcode, Bloodshed, Dev-C++	Arduino 0015
Programming Language Support	All C/C++ libraries	All AVR-g++ constructs
Open Source Projects?	Yes	Yes
Operating System Support	Windows/OS X/Linux	Windows/OS X/Linux

Table 10.2.2.1 Comparison of Programming Languages

The reasons for choosing an Atmel microcontroller are fully discussed in section 3.0, in addition there are several reasons Arduino is a good platform from a software perspective. Full operating system support alone makes the IDE very attractive as the team members regularly use Windows XP, Windows Vista, and Mac OS X, and thus team efforts towards programming and debugging will be easier compared to a development platform such as Xilinx and VHDL which are primarily supported on Windows. A minor factor is graphical user interface of the development software, which Xilinx and most C++ development suites sacrifice simplicity to provide a flexible and robust environment. The Arduino IDE is very simple and straightforward in comparison, and is pictured below in figure 10.1.2.1. A significant option that comes with Arduino is its ability to be extended using C++ libraries, meaning that if it is found a certain C++ function or library would enhance the software it can be included as needed. Much like C/C++ Arduino can also be compiled using a make file and command prompt, adding yet another degree of flexibility. As mentioned earlier, the overall function of the GuSu unit is relatively simple, and implementing the software should not require

strict object oriented or higher level programming that a simulation or complex system control would. Therefore the software shall be structured using specialized functions, variables, and parameters to run all functional systems, actions, and menus.

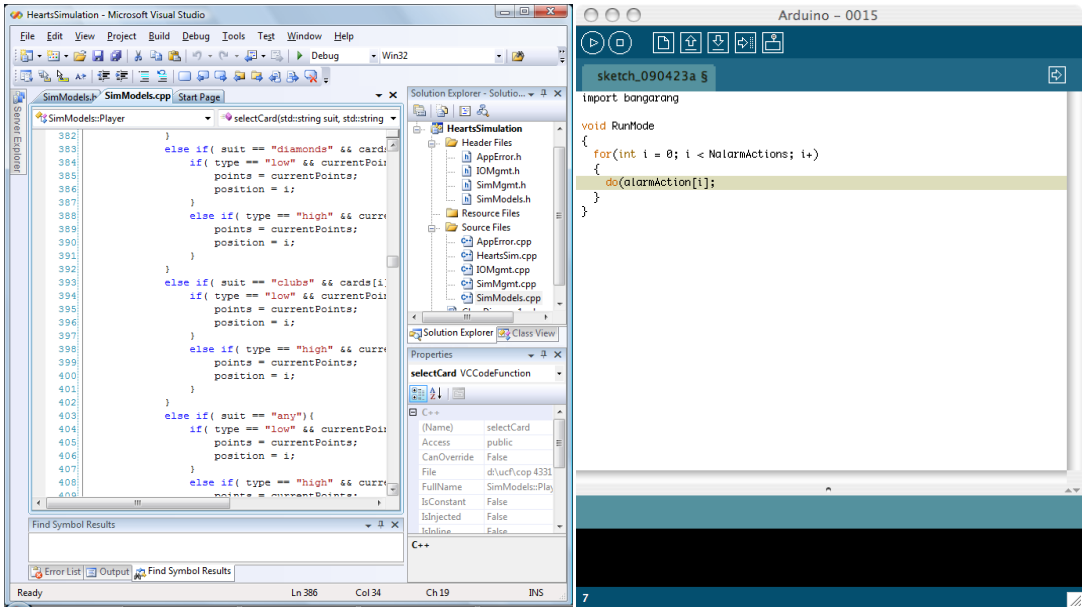


Figure 10.1.2.1: Side by side comparison of IDEs

10.2 Software Implementation

10.2.1 General Configuration

Two main functions will call and encapsulate all other functions; **SetMode** and **RunMode**. These are the only two modes the GuSu unit can be in once the system completes initialization, and as their names suggest, are the modes where the use changes settings and the mode where the GuSu alarm system is running, that is checking the time, checking for user presence in the sleeping area, and at alarm time if the user is present performing the user's preset alarm routines. Most alarm clocks when first turned on have default settings of midnight alarm and current time, and the alarm activated. This could be somewhat "dangerous" in a loose sense of the word given that if the GuSu is in **RunMode** and the clock returns the alarm time it will not let the user easily enter **SetMode** to silence it, and thus will force the user out of bed unless they point the infrared sensor away or find a similar fashion of becoming undetectable. But even escaping detection the user still cannot enter **SetMode** until either the alarm time span has passed or they hold the correct button for the chosen emergency silence time, currently chosen to be three minutes. To avoid this scenario entirely, however unlikely, the GuSu unit's startup routine will do the following; set clock to noon, set clock alarm to null, set all alarm and action durations to five

minutes, and finally enter **SetMode**, displaying the setting mode GUI menus on the LCD screen. To further ensure a correct initialization and setup for the first run the user will be forced to either choose a setting for or deactivate each available alarm action and the alarm time must be set.

A subroutine of the **SetMode** will be included that allows the user to test run the alarm actions following the routine they have specified, which can be stopped early, and a mode for testing the sensor positioning by signaling when it detects their presence using the on board alarm buzzer. Once the user has made their selections in SetMode, they are allowed to enter **RunMode** and the GUI on the LCD screen will switch to the running display as described in section 5.4. **RunMode** will also have a subordinate mode named **PlayMode**, which can be entered during any time not within the alarm time span, and will be overridden if alarm time is reached. Within this mode the user is allowed to play either FM radio or MP3 files from the on board SD card if it is present. This is not a major requirement as it does not facilitate waking the user and/or keeping them out of bed, but many similar devices, such as the iHome discussed before, provide the user this option, and implementation should not be overly complex. This feature does however fit along the design ideals for the GuSu system that improve the seamlessness and flexibility of the system. Before discussing the functions and modes in depth the global variables are named and defined in figure 10.2.1.1. Explicit pin numbering will also be saved as global variables for easy access and changing, and are laid out explicitly in table 3.3.1.

- **Int MP3data_pin**
  - Pin number assignment number for the MP3 decoder data line
- **Int MP3control_pin**
  - Pin number assignment number for the MP3 decoder control line
- **Int AUDIOmux_pin**
  - Pin number assignment for the audio line control mux
- **Int POWERmux_pin**
  - Pin number assignment for the power mux control
- **Int BUZZERmux_pin**
  - Pin number assignment for the buzzer mux control
- **Int SENSOR_pin**
  - Pin number assignment for sensor output
- **Int SDdata_pin**
  - Pin number assignment for the SD card slot data line
- **Int SDcontrol_pin**
  - Pin number assignment for the SD card slot control line
- **Int CLOCKin_pin**
  - Pin number assignment for the clock's in data stream
- **Int CLOCKout_pin**
  - Pin number assignment for the clock's out data stream
- **Int CLOCKalarm_pin**
  - Pin number assignment for the clock's alarm interrupt signal

- **enum modules { "MP3", "FM", "LOUD", "SOFT", "Coffee_XBEE", "Light_XBEE" }**
    - Enumerated list of modules available at alarm time, used for ordering alarm actions
- **int alarmActionOrder[6]**
    - Array that is filled in SetMode by the user's selection of order of actions during alarm time
- **enum weekdays { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday" }**
    - Enumerated weekdays
- **int current_WeekDay**
    - Integer 0-6 representing the current weekday, this value is incremented at midnight and reset to zero once 7 is reached
- **int current_SecondsFromMidnight**
    - This value is set over and over by requesting the current time from the onboard clock
- **int alarmTimes[7]**
    - Array of alarm times mapped to the weekdays in order
- **bool MP3_module**
    - Boolean variable whether the mp3 module will be used during alarm time
- **int MP3_order**
    - Order in alarm actions of the mp3 module
- **int MP3_duration**
    - Number of minutes mp3 files will be played
- **bool FM_module**
    - Boolean variable whether the mp3 module will be used during alarm time
- **int FM_order**
    - Order in alarm actions of FM module
- **int FM_duration**
    - Number of minutes FM sound will be played
- **bool LOUD_BUZZER_module**
    - Boolean variable whether the loud buzzer will be used during alarm time
- **int LOUD_BUZZER_order**
    - Order in alarm actions of the loud buzzer
- **int LOUD_BUZZER_duration**
    - Number of minutes the loud buzzer will sound for
- **bool SOFT_BUZZER_module**
    - Boolean variable whether the soft buzzer will be used during alarm time
- **int SOFT_BUZZER_order**
    - Order in alarm actions of the soft buzzer
- **int SOFT_BUZZER_duration**

    - Number of minutes the soft buzzer will sound for
- **bool XBEE_module**
    - Boolean variable whether the XBEE devices will be used during alarm time
- **int XBEE_order**
    - Order in alarm actions of XBEE activation
- **bool XBEE_connected**
    - Boolean variable indicating whether Zigbee devices have connected to the GuSu system's network
- **bool XBEE_atGetUp**
    - Boolean variable letting the user choose if they wish for Zigbee devices to activate in the order chosen only or as soon as they leave the bed
- **bool ON_BATTERY**
    - Boolean variable set to true when main power source goes low and the GuSu is running on battery. This variable is needed for decision making during alarm time and run time, explicitly explained in functions that check this variable.

Figure 10.2.1.1: Global Variable Definitions

The Zigbee module does not need a duration variable since devices that connect through Zigbee are preprogrammed internally to perform their own actions; they remain idle until a signal is received from the GuSu unit.

10.2.2 Overall Behavior

The **main** function will first perform any system test functions that are determined necessary next semester during building and once initialization of the system is complete enter **SetMode**, which will link the user to subordinate functions which handle changing settings for the GuSu alarm. These functions are; **ClockSet**, **AlarmSet**, **FMTunerSet**, **MP3Set**, **ToneSet**, **KeychainSet**, **XbeeSet**. Each of these Set functions is entered by selection of the user in the main **SetMode** menu. All Set functions will be structured to loop until the LEFT button signal is received, and will otherwise follow the user's navigation to select a sub function, each of which handle adjusting the variables associated with the device of the Set function. Exiting subordinate functions is handled the same way, detecting a LEFT signal, and will also loop until that signal is received. All set functions will behave in the following way; start with selection set to the highest option, loop until LEFT signal received, UP and DOWN signals will cause the selection to be incremented or decremented respectively, and a RIGHT signal will cause the program to enter the selected subordinate set function. Inside subordinate set functions UP and DOWN increment and decrement the related variable until a CENTER signal is received, causing the program to exit the loop and return to the main module Set function.

ClockSet will act differently from the other functions in that along with exiting on a LEFT signal, the input time will also be pushed out to the internal clock. This is

done because there is no guarantee the user will finish changing settings in less than a minute of choosing the clock time. It would be possible to track how much time elapses between choosing the clock time and exiting **SetMode** but this is adds unnecessary complexity and wastes resources. The **AlarmSet** function will also need explicit code that converts the user's abstract selections of 7-day, weekday/weekend, and every day for the alarm timing modes. In the case of 7-day the user's chosen times are saved to the **alarmTimes** array to the correct slot. For weekday/weekend the first five time slots are filled with the chosen weekday alarm times, while the last two slots are saved with the chosen weekend alarm time. If every day is selected then one time is saved to all spaces in the array. The **XbeeSet** function will require a **TestXbeeConnection** function which makes an attempt at connecting with any Zigbee devices in range and informs the user of what it successfully connected to. The final two functions in **SetMode** are **TestMode** and **BeginRun**. **TestMode** as mentioned above executes what would occur during alarm time if the user was detected in bed, this mode is exited if a LEFT signal is detected, and will output to the user what it is doing and for how long, while **BeginRun** calls **RunMode.** **RunMode** interprets all global variables one by one, first examining the alarm time settings and pushing the next desired alarm time to the internal clock. To perform this, the software must first check what the current day is, and if the alarm time for that day has passed or not. If it has not passed than the time for the current day is pushed to the clock, if it has passed then the alarm time for the next day is pushed to the clock. At the end of each alarm time span the next desired alarm time is retrieved from the **alarmTimes** array and pushed to the clock.

During alarm time spans power is sent to the sensor system and the output received is interpreted. Based on the current time and the time spans chosen for the various alarm actions the software will choose which action to take if the user is detected with the exception of Zigbee actions. For example given the scenario of the user setting the alarm time for the current day for 6 a.m., the FM tuner to play for fifteen minutes, followed by MP3 files for thirty minutes, and Zigbee devices to occur last with the wake up action turned off, the FM will go silent once the user rises from bed, if the user is detected in bed again at any time between 6:15 a.m. and 6:45 a.m. MP3 files will be played, and at 6:45 a.m. the Zigbee devices that successfully connect will be activated. At the start of each **RunMode** loop the current time will be requested from the clock and updated on the LCD screen, the upcoming alarm time will be displayed along with the chosen alarm actions. If the user has chosen to use the keychain tracker device a connection will be made with it once the user is no longer detected in the bed. The time the connection succeeds will be stored. Once the connection is lost the current time is compared to the time stored earlier and the difference is stored and averaged with previous values, along with the number of times compared being incremented by one. These values are used to estimate the time it takes the user to leave their home once they have left the bed and after a certain number of values being stored and averaged the option for "most sleep" will be available where the user can choose what time they wish to leave rather

than what time they wish to wake up, and the GuSU software will decide what time they need to be awoken based on that time and the time it takes them to leave.

RunMode will continue looping until a CENTER signal is detected, which will wait for the signal to continue over a time depending on if the current time falls within the alarm time span, if that wait time is reached **SetMode** will be called. The subordiante function of **RunMode, PlayMode**, will be entered if the UP signal is received over five seconds. This function let's allows use of the GuSu system as an FM radio or MP3 player, and is overridden if the alarm time is reached and the user is detected in bed. Overriding **PlayMode** is easily handled as when the alarm time saved to the clock is reached, the clock device chosen for the GuSu system automatically sends an interrupt signal. Figure 10.3.1.2 illustrates the overall flow of the GuSu system beginning at first turn on and running indefinitely.

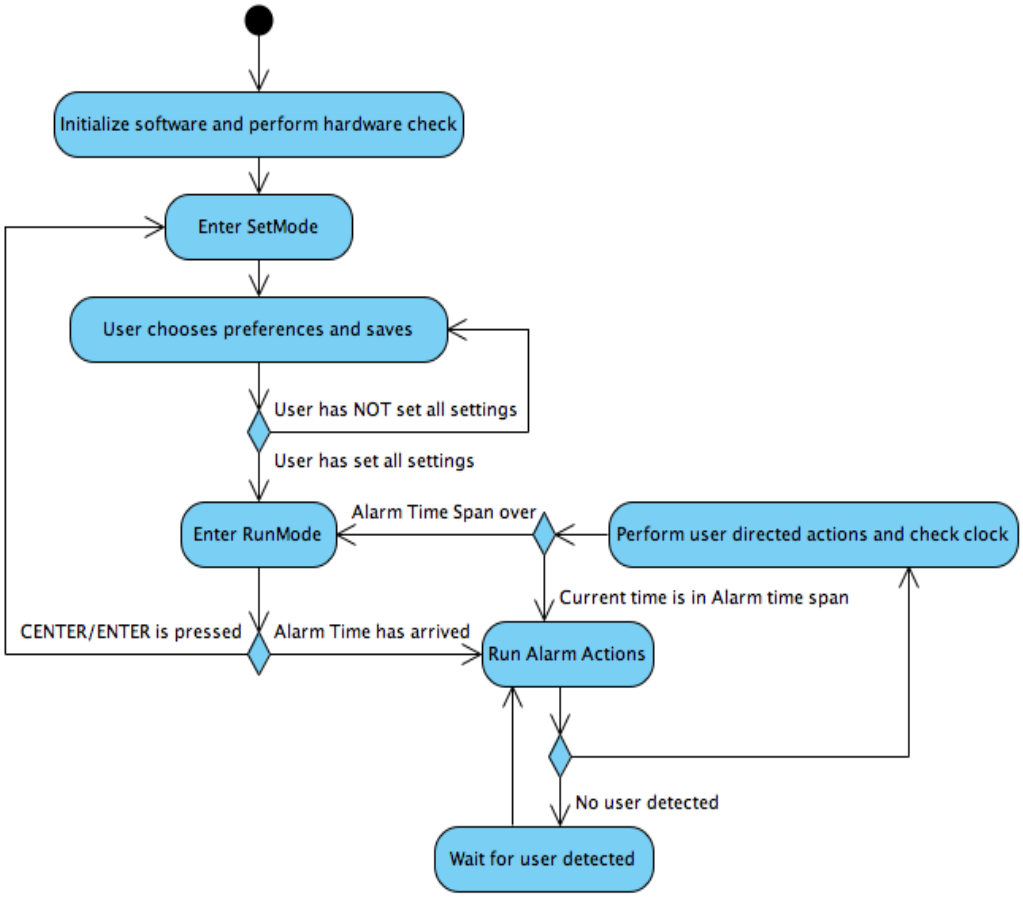

Initialize software and perform hardware check
Enter SetMode
User chooses preferences and saves
User has NOT set all settings
User has set all settings
Enter RunMode
Alarm Time Span over
Perform user directed actions and check clock
Current time is in Alarm time span
CENTER/ENTER is pressed
Alarm Time has arrived
Run Alarm Actions
No user detected
Wait for user detected

Figure 10.3.1.2: Overall Activity Diagram

10.2.3 Functions

10.2.3.1 RunMode Functions

The **RunMode** function itself acts as a loop and branch that continually checks the time and listens for user input. Depending on the current time, alarm time span, and user interaction **RunMode** will enter subordinate functions or call on **SetMode**. **RunMode**'s subordinate functions are laid out in name, purpose, and behavior here. All alarm action function exit and control options act similarly depending on how the function was entered. If the function was called because alarm time was reached then the user can only exit by holding down the CENTER/ENTER key and entering **SetMode**. If the function was called by **TestMode** the user can interrupt the function without delay and return back to **SetMode**. All alarm action functions also have a method internally that makes them continue running until the desired time span has elapsed. These functions must also check for user input so they can be interrupted by user interactions. All action functions will also be passed a parameter integer which informs them of how they were called.

10.2.3.2 Play_MP3

This function is called in one of three ways; alarm time is reached while the GuSu system is in **RunMode**, **TestMode** is called while the user is changing settings, or **PlayMode** is entered during runtime. This function handles sending data requests to the on board Secure Disk reader, and passing the data stream on to the MP3 decoder chip, and also controlling the chip. The function also sets the on board multiplexers control bits accordingly to direct power to the MP3 decoder chip and Secure Disk reader, along with directing the audio lines to accept only signal from routed from the MP3 decoder to the GuSu system's speakers. Play_MP3 differs from other alarm actions in that if the function was entered from **RunMode** during a non alarm time then the MP3 audio files are played under **PlayMode** which allows the user to advance or move back a song and information about the audio file playing is also displayed on the LCD screen.

10.2.3.3 Play_FM

Similar to Play_MP3, Play_FM must set multiplexer controls to route power to and audio from the FM tuner.

The FM tuner itself is relatively self contained and therefore does not require much interaction from the microcontroller directly. The FM station tuning will be handled externally to the microcontroller and so no LCD output or passing of control information needs to be passed on to the tuner by the microcontroller. This function can also be entered from **PlayMode**, allowing the user to have the FM radio playing normally, but will be overridden if alarm time comes.

10.2.3.4 Sound_Buzzer

Two multiplexers will be set by this function, one that passes power to the buzzer module, and one that selects either the loud or soft buzzer depending on what user preference has been set. If time allows during the build and test phase during the following semester an attempt will be made at adding the capability to slowly increase the volume of the loud buzzer. This will give the user the option for a rising alarm sound that for some can be a gentler way to wake up rather than sudden or loud noises. This will improve the GuSu system's ability to accomplish its overall goal of waking the user properly, which encourages the user to arise from bed and stay out of bed rather than simply react to a disturbance and return to slumber.

10.2.3.5 XBEE_cycle

This function will be called most often as it must act as the middleman between the GuSu system and all Zigbee devices in range. On startup a signal must be sent to the Zigbee module to create a network and start accepting connections from nearby devices. The XBEE_run function will send activation signals to Zigbee devices, accept and examine information returned from the Zigbee module, log disconnection times, and provide the necessary data for the microcontroller to display during **SetMode** and **RunMode** what devices have succeeded in connecting with the GuSu system. Disconnects, especially with the tracking device used for checking when the user has left their home, must also be reported to the GuSu system and handled accordingly.

10.2.3.6 Run_Alarm

The above functions are all called by this function, which is entered when the alarm time is reached and the GuSu system is in **RunMode**. Upon entering this method a check is made to the alarmActionOrder array, and the number given is used to decide which Boolean to check to see if that action was enabled by the user. If the action is enabled the related action function is called and executed until passing control back to the Run_Alarm function. Control is also given back if the user is no longer detected in the bed. If this occurs Run_Alarm will await either sensor signals to start alarm actions again, or user input to enter **SetMode** if the user holds down the CENTER/ENTER pushbutton long enough.

10.3 Software Testing

All software requires thorough unit testing to guarantee integrity and an implementation free of both runtime and logical errors. Unit testing will be written into and for all functions, and proof of concept versions will be written in C/C++ for external logic testing before attempting to make the software work directly with the microcontroller. If time allows a quick implementation of the menu system using Java will be created to make sure all logic and possible actions are exhausted to ensure there are no loop holes or bugs in the design of the menu

system. Finally, once all software is written, fully tested for logic errors, and uploaded to the microcontroller, rigorous use testing will be performed, hopefully with enough time to put the entire GuSu system through overnight implementation tests. No software can be considered 100% tested until it has been used in the field.

11.0 Printed Circuit Board

In order to attach all of the different modules and devices in this alarm clock together, it will be necessary to use a PCB (Printed Circuit Board). While all of the members of the group only have experience working with breadboards, it is not feasible to use one of these in the clock. They are large, bulky, and most importantly, do not allow the permanent soldering of the different devices. There are, of course, multiple methods of going about using PCBs in this project, however.

11.1 Research

The easiest method is often to outsource the actual creation of the PCB itself. This can be done through one of many companies online, all of which charge different prices for different services, but do roughly the same thing. They take a PCB schematic, use it to build a PCB, and return the PCB to the customer within a few days to a week. These PCB schematics are not the same as regular schematics, but that will be discussed later. The obvious advantages to this method of PCB development are the time and mistake savings. In other words, none of the group members have experience using a milling machine (the machine that can create the boards), it will take a lot of time for a couple of the members to learn the process of creating the board. There is also the time involved with getting certified to use the milling machine provided by UCF to the students, and while this time also helps to familiarize the users with the machine, it is not enough to make them truly competent users. This is also the cause of another time-drainer: mistakes. If any mistakes are made, it would render the board worthless and involve the purchase of another blank board.

It would likely still be worthwhile to become certified, however. There can be several reasons for this, but most importantly, it gives options in case there are mistakes in the design. If a design has a mistake in it, where a trace was made that shouldn't have been, or vice versa, it is possible that it could be fixed by using the milling machine at UCF rather than purchase another board. As will be seen below, while the boards are not ridiculously priced, it would get rather expensive to keep repurchasing the boards because of simple errors.

While the common companies available to produce PCBs do charge a premium to have them create the board for you, they are generally guaranteed to be done right. There are many different companies around, so each will be looked at to determine the feasibility of using that company. This applies to both the company manufacturing the board, and the company whose software will be

used to create the PCB schematic. These PCB schematics are known as 'Gerber files' [53]. These Gerber files are actually a collection of seven required files: Top Copper Layer (GTL), Top Soldermask Layer (GTS), Top Silkscreen Layer (GTO), Bottom Copper Layer (GBL), Bottom Soldermask Layer (GBS), Bottom Silkscreen Layer (GBO), and a Drill File [53]. The three-letter names in parenthesis in the previous sentence refer to the file extension of each file, respectively.

The first fabrication company that will be looked into is PCB123. A quick estimate of the needs of the group's board size would be about 8 X 10 inches, so that estimate will be used as a pricing estimate for the different companies as of now. Of course, when the final PCB schematic is completed, the choice of company could be changed if the final dimensions are significantly different, and would change the choice of PCB producer. As mentioned earlier, regardless of the software chosen, the hardware development uses the same files. Because it is standardized, any company's software can be used, and the files sent to a different PCB producer, if desired. For 2 PCB boards of 8 X 10 inches, PCB123 will create them for $81.00 [61]. This includes free ground shipping in the US, and is only including a 2-layer board with no soldermask or silkscreen. These layers are considered rather unnecessary for prototype boards, which are what the group is looking for. The minimum trace width is .007 inches, and the minimum space width is not available online. This could be important to find out if this manufacturer were chosen, as some parts (such as the Si4704 FM Tuner) are extremely small, and need very small trace and space widths.

PCB123 also offers free software available for download to create the PCB schematics. This software is nice in that it comes with a parts library of over 100,000 parts free, as well as an over 100 page user manual, and a part placement guide, and many other features. It will definitely be considered when selecting the program to use.

Another fabrication company that stands out from the rest is ExpressPCB. Using the same criterion as with the previous board, a price estimate is important. From their website, they use a formula to determine the final price, rather than tables. The formula they use is: "$55 + ($0.65 * NumberOfBoards * BoardAreaInSquareInches) + ($1.00 *NumberOfBoards) + Shipping" [62]. With ordering 2 boards, with no silkscreen or soldermask layers, and a shipping fee of $9.85 for 2-Day air (the only readily available shipping quote; it may be possible to get it cheaper with slower delivery), this comes out to $171. If only one board were needed, this total would be $118. This is significantly higher than PCB123's price, although ten of it may be removable in unnecessary shipping costs.

This fabrication company also offers some free software to design the PCBs with. This software does not come with the vast library that PCB123's software does, but it does come with a basic schematic editor as well, ExpressSCH. This is

extremely handy. Using schematic software that is more similar to pSPICE is enticing, as the group has plenty of experience in using that software. The schematic, when completed, can be linked into when using the PCB software, ExpressPCB. The pins are all linked to, and if the naming is used consistently, the ExpressPCB software will show the user what pins need to be connected to what other pins. Now, the software has an automated feature that may be worth looking into that will automatically connect all the different traces, but with a board as complex as this one, it may be more fool-proof if the group decides to manually trace the lines in the program.

A third option for software to use for the PCB board is Eagle. This program is pretty top-notch, and multiple good tutorials were found for it online. A great benefit to the program is that there are huge libraries, containing most ICs, available to download. However, the program is expensive, running 747 dollars for 1 user to make a board about the size this group would need. There is a freeware version offered, but it is limited to 4 x 3.2 inch board dimensions. The final board that will be used for the alarm clock will most certainly exceed these dimensions.

11.2 Implementation

The software chosen was the ExpressSCH, along with the ExpressPCB. These were chosen for their ease of use, helpful, quick tutorial [63], the fact that it was free, and most importantly, the connection between the schematic and the PCB programs for error-checking. Each group member downloaded the free programs, so that schematics created on one machine would be able to be seen, and checked, on the other member's computers.

First, the tutorial from www.usna.edu was run through, using both the ExpressSCH and the ExpressPCB software [63]. Manipulating both pieces of software became much easier after this. Next, the master schematic of the Gusu Alarm clock was created. It was important to label the pins on this schematic so that they would match the pins on the pieces of hardware itself. This schematic can be seen in the final design summary. Most of the parts had to be manually created. It was very important that each part, including the basic miscellaneous parts such as resistors and capacitors, each had their own part names. The part names are required for linking to the PCB editor, the PCB editor had to know what pins on each specific part (which was known by its part name) was supposed to connect to which pins on another part. Also, for simplicity, most of the signals were not passed by physical lines; rather, they were tagged with labels, and any two lines tagged with the same label were considered to be electrically connected. This physical schematic is also attached in the Appendix.

The PCB editor was used to create a final PCB schematic. This process was much more involved than the first step. Each part that would be soldered onto the board had to have its true physical dimensions known, and a figure had to be created in the program for it. Although the editor did have parts such as common

DIP ICs available (e.g. DIP 20, DIP 28, etc.), it did not have parts such as the buzzer, or the multiplexor, in its library, and these had to be created from scratch. If these did not match the hardware perfectly, there would be problems when the time came to install them. A list of how each piece of hardware would interface with the board was created, and is recreated below in Table 11.2.1.

Hardware	Connection Method
Microcontroller	DIP 40
MP3 Decoder	DIP 28
Clock	DIP 16
FM Tuner	DIP 18
Digital to Analog converter	DIP 8
Zigbee	DIP 20
Op-Amp	Dip 8
5V regulator	TO-220
Variable regulator	TO-220
PIR	Wires to PCB
Buttons	Wires to PCB
SD Card Reader	Wires to PCB
Batteries	Wires to PCB
LCD	Wires to PCB
Multiplexor	Custom
Buzzer	Custom

Table 11.2.1: The main pieces of hardware in the Gusu clock, and the method that each would use to interface with the PCB.

Next, each part was placed. They were placed in a manner to reduce the crossing of lines as much as possible, and to keep signals from interfering (such as the FM tuner and the Zigbee device). Of course, with the quantity of signal lines needed, it was impossible to keep any lines from crossing. While the bottom layer of the PCB is a ground layer, it was possible to trace out lines from the ground layer as well to reduce the problem. However, even using both layers, it was impossible to cross all of the signal lines. It was decided that a few wires would have to be used to cross over lines, although the number to be used was kept to a minimum.

It was decided to make a majority of the signal traces .01” in width, as that is sufficient to pass 300mA of current through with no problems [63]. However, the power traces were chosen to be .025” in width, which can handle a current load of 1A [63]. Of course, this is overkill, but as in any design, it is better to have a system that is rated to handle more of everything than is expected.

Using netlisting (the handy tool that shows what each pin should connect to, when the PCB is linked to a schematic), the parts were connected as planned. The ground plane was created, and each pin was double checked. The group

reviewed the design, and agreed to it after a couple of minor changes. The final design is shown below in Figure 11.2.2, and a full-size copy is in the Appendix.

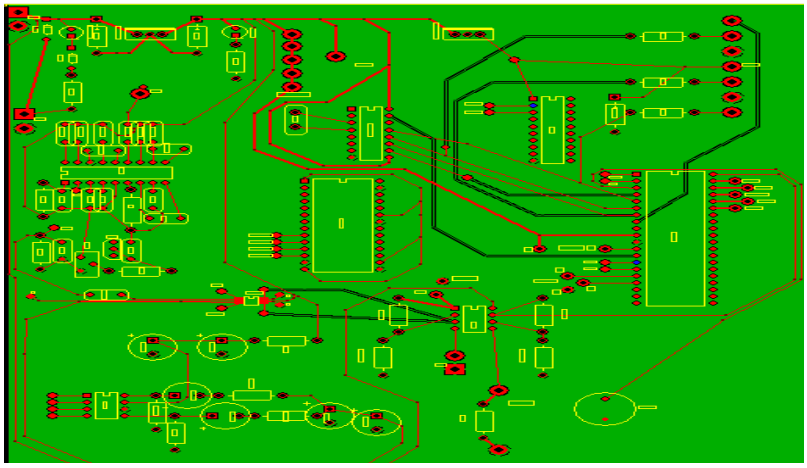


Figure 11.2.2: The final PCB Design of the Gusu Clock. The PCB was created using ExpressPCB.

12.0 Budget

Dealing with budgets and the financials of a project is an important role in the design process. Budgets are normally the first things discussed when companies talk about the development of software and hardware projects. If the budget cost is too high and the projected revenues from the developed project cannot surpass the initial budget outlay chances are companies will not go ahead with that project. The same question must be asked for the GuSu prototype. Is this design cost feasible? The GuSu project team definitely thought that the overall design is cost feasible for the purpose it is intended for. The team would have to purchase all components and other necessary devices for the prototype to perform. Each team member asked their various companies of employment for a sponsorship on the project but was unable to get a sponsor to pay for the prototype. The team members thought of asking the local faculty and professors for a sponsor but assumed the same outcome as the team member's employers. One of the reasons the team thought of for not being able to obtain a sponsorship was the poor economy. Businesses and faculty members of UCF are having a hard enough time paying their own costs and paying for student's projects is not on their list of expenditures. Although this was a disappointment the GuSu team members were able to get student discounts on some of the components purchased. The cost for this project is coming directly out of the team member's pockets so these discounts helped to curb the cost of supplies and components. Seen in table 12.1 is the planned budget, this is what the team originally thought of purchasing for the GuSu prototype. Table 12.2 has information on the development budget, these are the items the team will be using to complete the project but already have for free. Lastly is the actual budget in table 12.3 with all of the components and devices that were bought.

Components	Pricing
LCD display	$50.00
XBee Modules	$40.00
Housing/Case Supplies	$20.00
Coffee Machine	$20.00
Microcontroller	$15.00
FM Tuner	$5.00
SD Card	$5.00
PIR Sensors	$10.00
MP3 Decoder	$15.00
SD Card Reader	$20.00
Speaker	$5.00
PrintedCircuitBoard	$100.00
Miscellaneous	$20.00
TOTAL	$325.00

Table 12.1: Planned Budget

Components	Pricing
Electronic Breadboard	$40.00
Soldering Gun	$50.00
4DSystemsSoftware	Free
ArduinoSoftware	Free
TOTAL	$90.00

Table 12.2: Development Budget

Components	Prototype Cost	Production Cost
uOLED-160-G1 LCD Display	$99.95 (1)	$99.95
Atmel ATmega644-20PU	$7.87 (1)	$7.87
Sanguino Development Kit	$25.00 (1)	$25.00
XBee Modules	$23.00 (2) = $46.00	$23.00
Basic Coffee Machine	$20.00 (1)	$30.00
Arduino Development Kit	Free, Josh	$25.00
ATmega168	Free, Josh	$5.49
Housing/Case Supplies	$25.00 (1)	$20.00
SD Card	$4.50 (1)	$3.00
SD Card Socket	$3.95 (1)	$3.00
DS1305 Clock Timer	$5.06 (1)	$5.00
TDA7000 FM Tuner	$7.00 (1)	$6.00
PIRD203B Passive Infrared Sensor	$1.90 (2) = $3.80	$1.75
PIRd203S Directional Infrared Sensor	$1.90 (2) = $3.80	$1.75
Fresnel Lens	$0.35 (5) = $1.75	$0.30
PIR Sensor Module	$6.90 (1)	$6.90
Compact PIR Sensor Module	$7.40 (1)	$7.40
CS9803 Infrared Induction Control	$0.90 (3) = $2.70	$0.90
LP8072 PIR Sensor	$0.60 (3) = $1.80	$0.60
M7612 PIR Controller	$0.90 (3) = $2.70	$0.90
STA013 MP3 Decoder	$6.90 (2) = $13.80	$12.00

28 Pin SOIC Adapter	$0.80 (2) = $1.60	$0.75
LM7805 5V Regulator	$0.51 (1)	$0.50
DE-SWADJ 3.3V Variable Regulator	$15.00 (1)	$12.00
WST-1205S Buzzer	$1.81 (1)	$1.80
LM1458 Op-Amp	$0.50 (1)	$0.50
12V Wall Wart	Free, Andrew	$12.00
EAS-4P15SA Speaker	$4.32 (1)	$4.00
TS5A23159DGSR Mux	$0.81 (1)	$0.81
Printed Circuit Board	$80.00 (1)	$60.00
Miscellaneous	$10.00 (1)	$5.00
TOTAL	$403.53	$383.17

Table 12.3: Actual Budget

13.0 Final Design Review & Integration Test

The purpose of this section will help clarify all of the research and implementation that went into each component. To make the reader understand more about how the GuSu is designed to function a walkthrough of each component and how they interact with one another will be discussed.

The main component which will control all other modules will be the microcontroller. All of the other components will be directly involved with this device. The LCD display will also be another key component with the GuSu design. It will connect to the microcontroller and be the way the prototype interacts with the user through a menu system and buttons to allow the prototype to perform the various functions. Another electronic component that will make the design unique is the PIR sensors. These sensors will be positioned somewhere near the users bed, possibly on the side or above. This is the component that will allow the prototype to test whether the user is in bed or not and will act accordingly by turning on or off the alarm buzzer and speaker. To wake the user from sleep the GuSu design will use audio output in the form of a speaker and a buzzer. Interacting with the audio output is also the SD Card and the MP3 decoder. The SD Card will primarily be used for storage of files such as music and audio files. This will connect to the MP3 decoder which can then decipher the audio files and output them to the speaker. Included with the audio components will also be a FM Tuner so the user has the option of listening to FM radio. One of the last major components is the addition of the wireless ZigBee module which will interact with the Coffee Machine. This will allow the Coffee Machine to brew its coffee at the time when GuSu alarm sounds.

The external casing design will probably be the easiest part of the design to make. This will be the housing for all of the electronic components to fit neatly

together. The LCD display will be shown through the front of the housing and have push buttons near the display to control the menu system. The power and battery backup systems will also be involved with this part of the design. To power the device it will be plugged into a standard AC wall outlet and also have a battery backup system. The purpose of the backup system is if a power outage occurs the GuSu alarm clock will still have power to wake the user up in the morning. Inside of the housing the components will be connected to a Printed Circuit Board to have control of the devices. Shown on the next page, in Figure 13.1, is a schematic of the Printed Circuit Board for the GuSu design showing all of the components the team has selected to work with one another.

The final integration test of the project will be the most important tests to perform to make sure the prototype functions correctly. The final test will most likely be a culmination of all of the other tests combined. Working in a logical and sequential order the main components will first be tested and then the other component modules will be added on step by step. The team will be using the UCF labs and breadboards to experiment with all of the components. The first device to be tested will be the microcontroller since this is the heart of the prototype. The other modules will be added onto the microcontroller one by one. After one device successfully works with the microcontroller the team will wire and solder up that component and this will continue until all devices are tested and then connected. After all contacts are made and are working properly the software testing will begin. Software coding will be the last tests made on the devices. The code that will be implemented onto the GuSu prototype will be the control factor for all of the components. These devices can be connected with ease but without software to control the design the final outcome would be useless. The parts that have been ordered for the prototype will be here in time for Senior Design II which the GuSu team members will be taking in the summer of 2009. This final class will be the ultimate goal for the GuSu team and the entire semester will be spent testing and configuring the GuSu prototype.

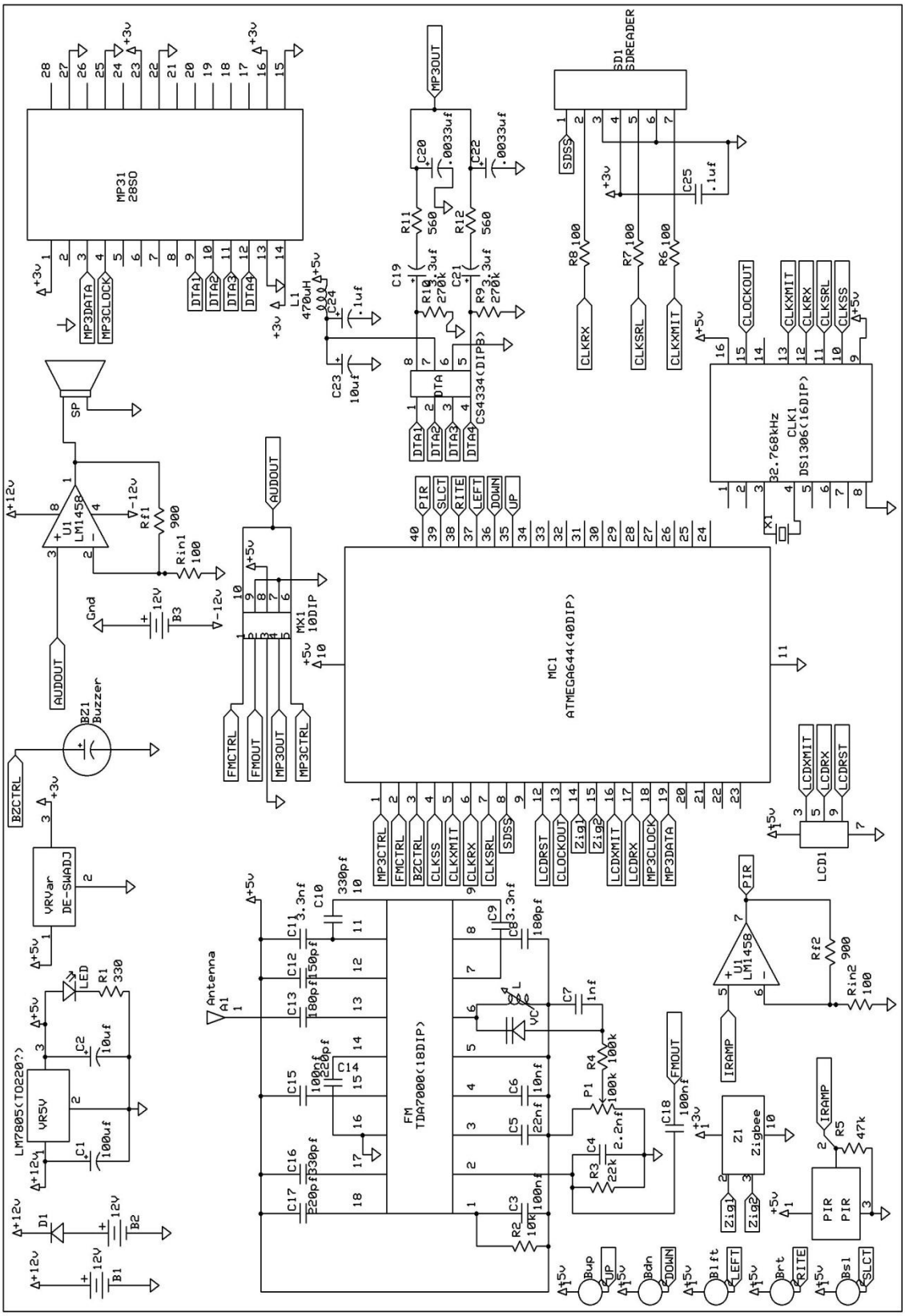


Figure 13.1: Schematic of the Printed Circuit Board with components.

14.0 Conclusion

With the American workforce being so high paced in this day and age there will be a need for a new and improved alarm clock to help them with their daily lives. Many people are not satisfied with their current alarm clock setup and are looking to enhance their present design with better features and a more improved user interface. The GuSu prototype will challenge this need with a better design implementation and more efficiency.

The goal of the GuSu development team will be to dive into the electronic field of alarm clocks and decide what users really want and need. Most modern alarm clocks on the market today are simple and affordable for the standard user. The GuSu prototype will attempt to revamp these simple alarm clocks with a more proficient design geared towards a wide genre of users. As stated in the executive summary, there are many problems with the amount of sleep each person gets every night. The GuSu prototype will ensure that everyone will get a good night's rest and will guarantee that the user is awakened whenever they need to be. The two core types of people the GuSu prototype will attempt to persuade in utilizing the device will be people in school and people that are in the workplace. The GuSu development team knows from personal experience how crucial it is to waken from sleep and get started on the daily activities. Not waking up in the morning seems harmless enough but it can cause major consequences down the road.

The GuSu prototype that the team has developed will help ease these troubles. The initial budget costs of the design will be quite pricey but if the design was manufactured on a mass scale the overall cost to the average user will drop significantly. The components used in the GuSu prototype are standard devices that can be bought off the internet from companies that specialize in the manufacture of these specific devices. One good aspect about the GuSu prototype is its capability of adding new features and devices with relative simplicity. One big feature that that can be upgraded to the average user will be the Menu system. To add new devices it will be simple to add them to the microcontroller but the software would have to be implemented to control it.

Keeping in mind all of the features and usability the GuSu prototype has to offer there will be many benefits that outweigh the costs of actually building the prototype. The prototype will be safe enough to be used in any location of the user's home and users of all ages will be able to operate the prototype easily with simple instructions. The GuSu design will also attempt to challenge the users of the prototype to come up with new ideas and features. In conclusion, the GuSu prototype will have a lasting impact on the feasibility of new alarm clock designs in the future and will help pave the way for new and improved designs.

APPENDIX A: Works Cited

[1] http://www.sleepfoundation.org/site/c.huIXKjM0IxF/b.3934129/k.31D9/Poll_Stats.htm, Accessed 4-8-09

[2] http://eecs.ucf.edu/seniordesign/su2007fa2007/g04/Site/Welcome.html, Accessed 3-29-09

[3] http://me.columbia.edu/seniordesigns/2007/800/ultimate.html, Accessed 3-29-09

[4] http://eecs.ucf.edu/seniordesign/sp2007su2007/g09/, Accessed 3-29-09

[5] http://www.atmel.com/dyn/resources/prod_documents/doc8011.pdf
ATmega644, Accessed 3-25-09

[6] http://sanguino.cc/hardware
ATmega644, Accessed 3-25-09

[7] http://arduino.cc/en/Main/Hardware
ATmega164p, Accessed 3-25-09

[8] http://www.howstuffworks.com/question124.htm
Accessed 4-1-2009

[9] http://www.hotsolder.com/articles/index.php?page=stupid-op-amp-trick
Accessed 3-31-2009

[10] http://www.microwaves101.com/encyclopedia/receivers_superhet.cfm
Accessed 3-1-2009

[11] http://www.semiconductor-sanyo.com/easy_radio/index.htm
Accessed 3-2-2009

[12] https://www.silabs.com/products/mcu/Pages/USBFMRadioRD.aspx
Accessed 3-7-2009

[13] http://electronics-diy.com/TDA7000_FM_Receiver.php
Accessed 4-2-2009

[14] http://www.users.bigpond.com/cool386/tda7000/tda7000.html
Accessed 4-2-2009

[15] http://www.vlsi.fi/en/products/vs1002.html, Accessed 4-7-09

[16] http://www.futurlec.com/Mini_MP3.shtml, Accessed 4-11-09

[17] http://www.pjrc.com/tech/mp3/sta013.html, Accessed 4-11-09

[18] http://www.rohm.com/products/lsi/sound/usb_host_audio/, Accessed 4-11-09

[19] http://microcontrollershop.com/product_info.php?products_id=1422, Accessed 4-11-09

[20] http://www.sparkfun.com/commerce/product_info.php?products_id=7832, Accessed 4-5-09

[21] http://www.vlsi.fi/en/products/vs1053.html, Accessed 4-7-09

[22] http://www.vlsi.fi/en/products/vs1001.html, Accessed 4-7-09

[23] http://plex.us/outbursts/dc_memory.html, Accessed 4-20-09

[24] http://www.ghielectronics.com/product/2, Accessed 4-9-09

[25] http://www.sparkfun.com/datasheets/Prototyping/SD-Socket-PP-14446.pdf, Accessed 4-9-09

[26] http://www.sparkfun.com/commerce/product_info.php?products_id=8698, Accessed 4-9-09

[27] http://www.cs.ucr.edu/~amitra/sdcard/Additional/sdcard_appnote_foust.pdf, Accessed 4-9-09

[28]http://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html, Accessed 4-10-09

[29] http://www.britannica.com/EBchecked/topic/343093/liquid-crystal-display, Accessed 3-13-09

[30] http://www.4dsystems.com.au/prod.php?id=29, Accessed 3-13-09

[31] http://www.futurlec.com/, Accessed 3-28-09

[32] http://www.surplussales.com/Switches/SWPushB-1.html, Accessed 4-15-09

[33] http://www.digikey.com/, Accessed 4-18-09

[34] http://www.ihomeaudio.com/products.asp?product_id=10309&dept_id=1006, Accessed 4-2-09

[35] http://www.vat19.com/dvds/flying-alarm-clock.cfm, Accessed 4-9-09

[36] http://www.tekscan.com/flexiforce/flexiforce.html, Accessed 4-19-09

[37] http://www.parallax.com/Store/tabid/60/Default.aspx, Accessed 4-19-09

[38] http://www.futek.com/, Accessed 4-20-09

[39] http://www.imagesco.com/sensors/force-sensors.html, Accessed 4-20-09

[40] http://www.sparkfun.com, Accessed 3-25-09

[41] http://www.efx-tek.com/topics/pir.html, Accessed 4-17-09

[42] http://www.futureelectronics.com/, Accessed 4-13-09

[43] https://www.fallingpixel.com/product.php/6877, Accessed 4-14-09

[44] http://bluetooth.com/Bluetooth/Technology/Works/Architecture__Radio.htm
Accessed 3-1-2009

[45] http://zensys.org
Accessed 3-3-2009

[46]
http://www.zigbee.org/imwp/idms/popups/pop_download.asp?contentID=10547
Accessed 3-1-09

[47] http://www.digi.com/technology/rf-articles/wireless-zigbee.jsp
Accessed 3-10-09

[48] http://www.digi.com/pdf/ds_xbeezbmodules.pdf
XBee ZB Pro, Accessed 3-10-09

[49]
http://www.energysavers.gov/your_home/appliances/index.cfm/mytopic=10040
Department of Energy, Accessed 4-8-09

[50] http://www.atmel.com/dyn/resources/prod_documents/doc2543.pdf
ATtiny2313, Accessed 4-3-09

[51] http://www.c-max-time.com/downloads/getFile.php?id=514
WWVB Clock IC, Accessed 3-30-09

[52] http://datasheets.maxim-ic.com/en/ds/DS1305.pdf
DS-1305 RTC, Accessed 3-30-09

[53] http://www.sparkfun.com/commerce/tutorial_info.php?tutorials_id=57
Accessed 3-10-2009

[54]http://www.amazon.com/s/?ie=UTF8&keywords=wall+wart&tag=googhydr-20&index=electronics&hvadid=2565704181&ref=pd_sl_352wjrl16u_b
Accessed 3-11-2009

[55]
http://store.mp3car.com/AC_DC_Power_Brick_120_Volt_AC_to_12V_DC_p/pwr-028.htm
Accessed 3-11-2009

[56] http://www.national.com/mpf/LM/LM78M05.html
Accessed 3-11-2009

[57]
http://www.digikey.com/scripts/dksearch/dksus.dll?KeywordSearch?Keywords=LM3350MM&vendor=14
Accessed 3-12-2009

[58] http://www.dimensionengineering.com/DE-SWADJ.htm
Accessed 3-12-2009

[59] http://www.techlib.com/reference/batteries.html
Accessed 3-12-2009

[60] http://www.arduino.cc/, Accessed 3-10-09

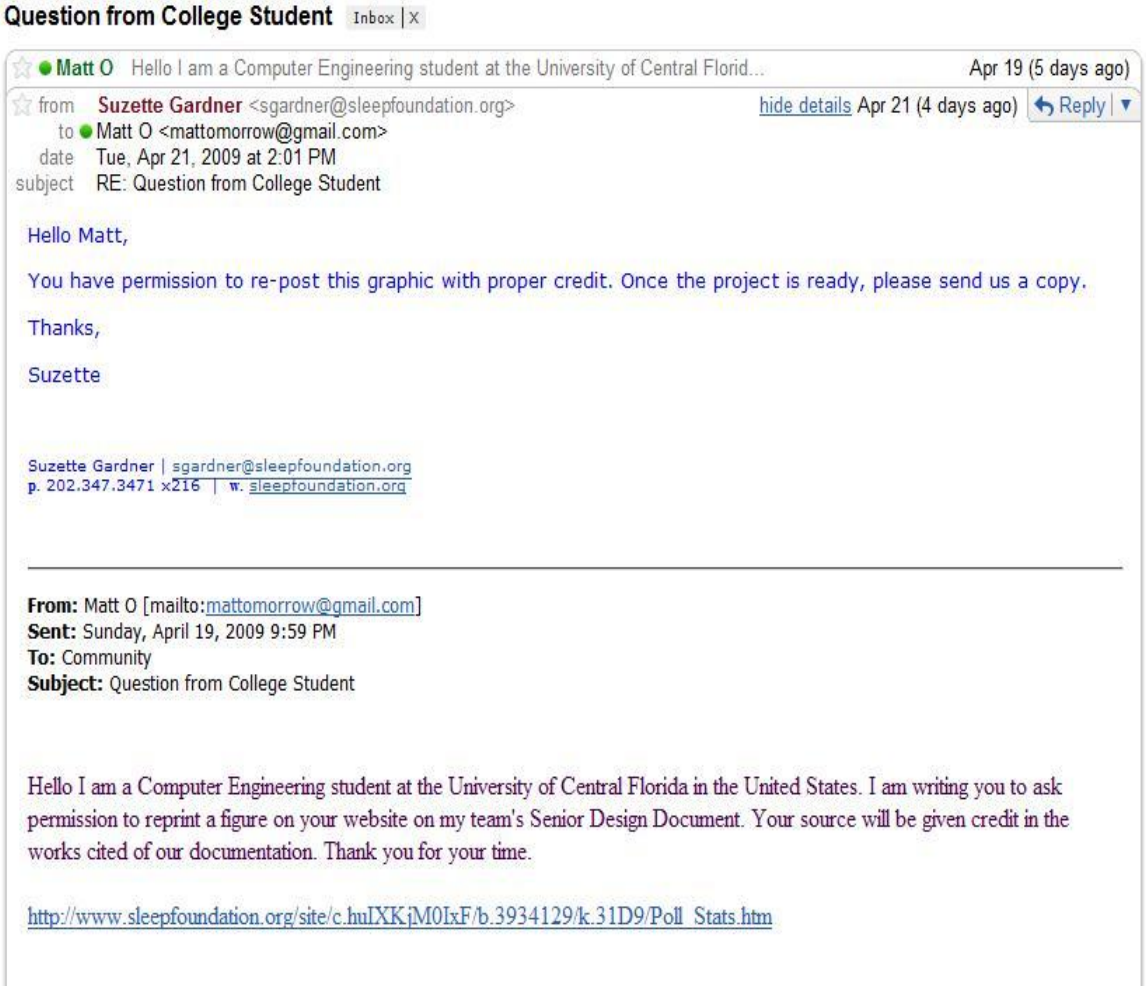
[61] http://www.pcb123.com/pcb123pricing.php
Accessed 3-27-2009

[62] http://www.expresspcb.com/ExpressPCBHtm/SpecsStandard.htm
Accessed 3-27-2009

[63] www.usna.edu/EE/ee241/EXPRESS%20PCB%20TUTORIAL.doc
Accessed 3-29-2009

APPENDIX B: Emails

Figure 1.0.1



Question from College Student Inbox | X

Matt O Hello I am a Computer Engineering student at the University of Central Florid... Apr 19 (5 days ago)

from **Suzette Gardner** <sgardner@sleepfoundation.org> hide details Apr 21 (4 days ago) Reply

to Matt O <mattomorrow@gmail.com>
date Tue, Apr 21, 2009 at 2:01 PM
subject RE: Question from College Student

Hello Matt,

You have permission to re-post this graphic with proper credit. Once the project is ready, please send us a copy.

Thanks,

Suzette

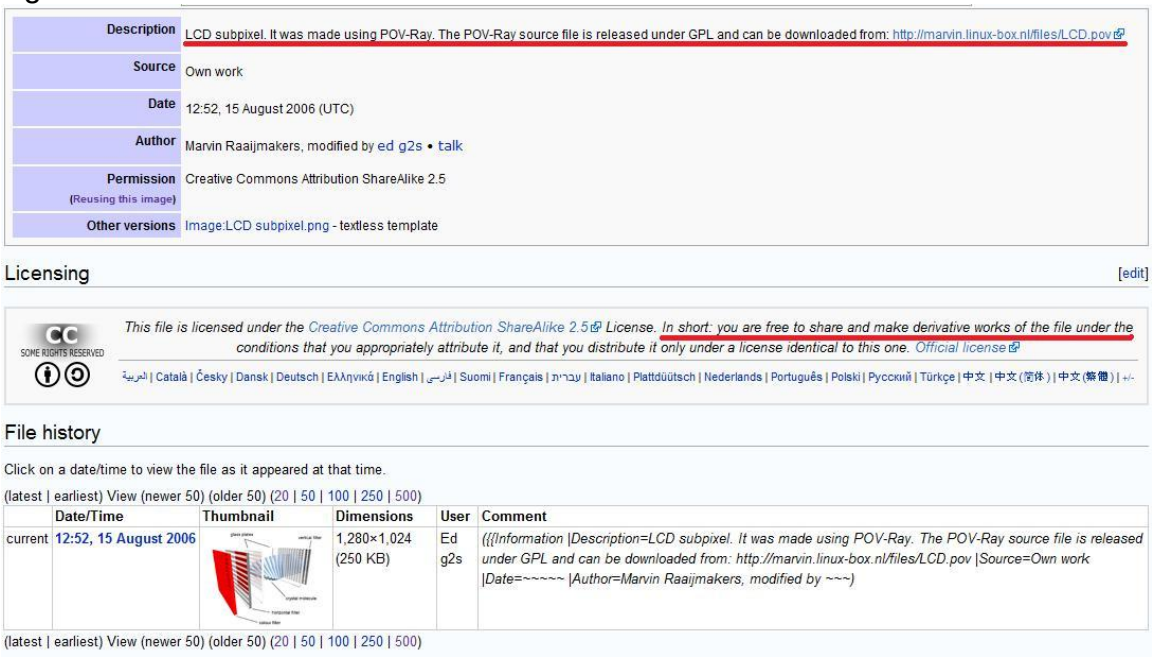
Suzette Gardner | sgardner@sleepfoundation.org
p. 202.347.3471 x216 | w. sleepfoundation.org

From: Matt O [mailto:mattomorrow@gmail.com]
Sent: Sunday, April 19, 2009 9:59 PM
To: Community
Subject: Question from College Student

Hello I am a Computer Engineering student at the University of Central Florida in the United States. I am writing you to ask permission to reprint a figure on your website on my team's Senior Design Document. Your source will be given credit in the works cited of our documentation. Thank you for your time.

http://www.sleepfoundation.org/site/c.huIXKjM0IxF/b.3934129/k.31D9/Poll_Stats.htm

Figure 5.2.1.1



Description	LCD subpixel. It was made using POV-Ray. The POV-Ray source file is released under GPL and can be downloaded from: http://marvin.linux-box.nl/files/LCD.pov
Source	Own work
Date	12:52, 15 August 2006 (UTC)
Author	Marvin Raaijmakers, modified by ed g2s • talk
Permission (Reusing this image)	Creative Commons Attribution ShareAlike 2.5
Other versions	Image:LCD subpixel.png - textless template

Licensing [edit]

This file is licensed under the Creative Commons Attribution ShareAlike 2.5 License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license identical to this one. Official license

العربية | Català | Česky | Dansk | Deutsch | Ελληνικά | English | فارسی | Suomi | Français | עברית | Italiano | Plattdüütsch | Nederlands | Português | Polski | Русский | Türkçe | 中文 | 中文(简体) | 中文(繁體) | +/-

File history

Click on a date/time to view the file as it appeared at that time.

(latest | earliest) View (newer 50) (older 50) (20 | 50 | 100 | 250 | 500)

	Date/Time	Thumbnail	Dimensions	User	Comment
current	12:52, 15 August 2006		1,280×1,024 (250 KB)	Ed g2s	*{{Information |Description=LCD subpixel. It was made using POV-Ray. The POV-Ray source file is released under GPL and can be downloaded from: http://marvin.linux-box.nl/files/LCD.pov |Source=Own work |Date=~~~~~ |Author=Marvin Raaijmakers, modified by ~~~}}*

(latest | earliest) View (newer 50) (older 50) (20 | 50 | 100 | 250 | 500)

Figure 5.2.1.2

Question from College Student

from Matt O'Morrow <mattomorrow@gmail.com> hide details 3:56 PM (48 minutes ago) Reply

to media@howstuffworks.com
date Sun, Apr 26, 2009 at 3:56 PM
subject Question from College Student
mailed-by gmail.com

Hello I am a Computer Engineering student at the University of Central Florida in the United States. I am writing you to ask permission to reprint some figures on your website on my team's Senior Design Document. Your source will be given credit in the works cited of our documentation. Thank you for your time.

http://static.howstuffworks.com/gif/oled-cell.gif

Figures 5.2.1.3, 5.2.2.1, 5.2.2.2, 5.2.3.1, 5.2.3.2

Hi Matt,

You can use the figures as long as you give proper references of the product and the company in your documentation.

Cheers

Muhammad Bilal

4D Tech Support

email: support@4dsystems.com.au

Web: http://www.4dsystems.com.au

4DGL: http://www.4dsystems.com.au/developers

Forum: http://www.websitetoolbox.com/mb/4d

From: Matt O'Morrow [mailto:matt@hudat.com]
Sent: Friday, March 13, 2009 4:12 AM
To: support@4dsystems.com.au
Subject: uOLED-160-G1 LCD Display

Matt OMorrow

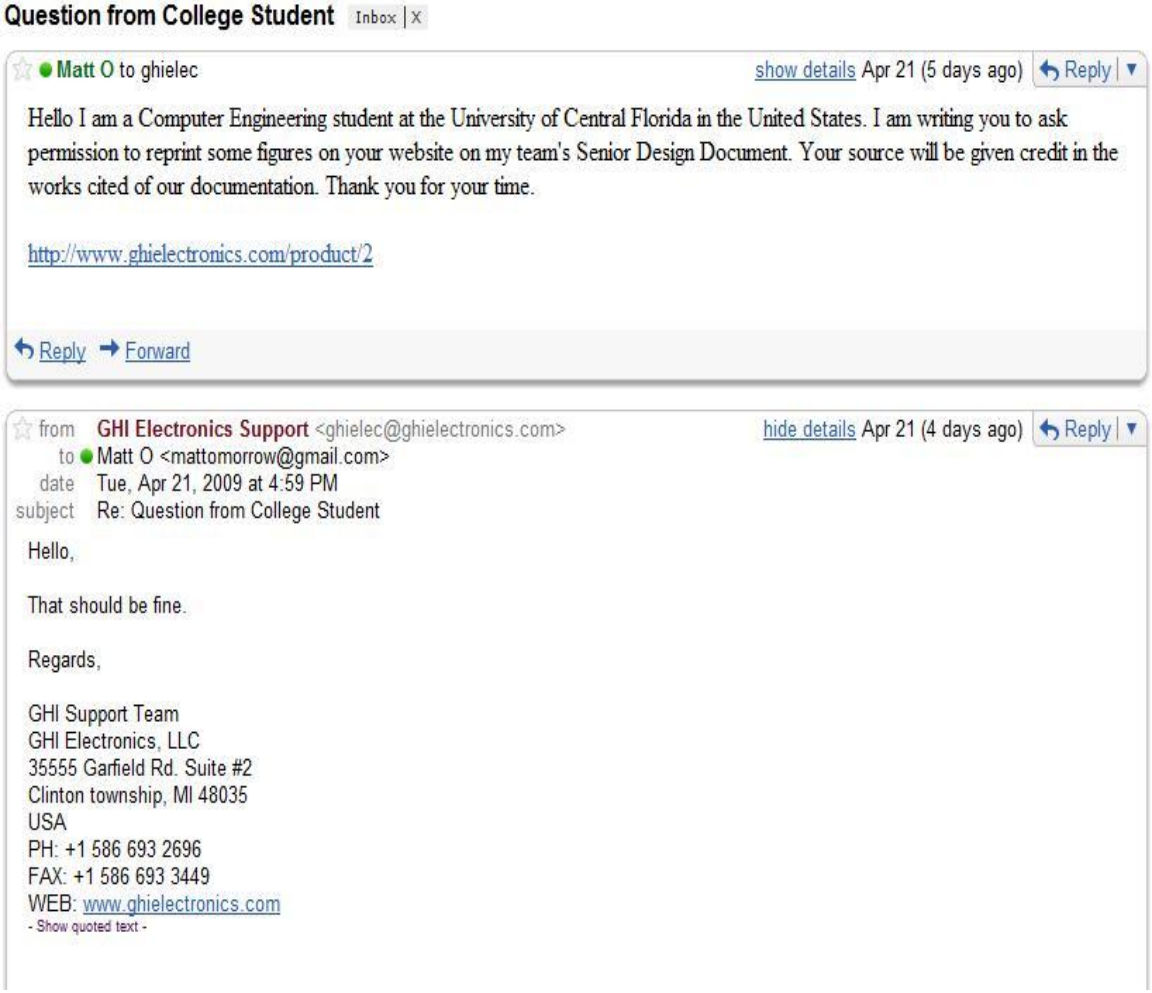
United States

matt@myacc.net

Hello, I am an engineering student at the University of Central Florida in the United States. We are working on a Senior Design project and will be using your uOLED-160-G1 LCD display. I am writing to ask for permission to use figures you have in your PDF files to add to our documentation.

uOLED-160-G1-Mech.pdf
uOLED-160-G1_Users_Manual_Rev1.0.pdf
uOLED-Connections.pdf

Figures 4.2.4.1.1, 4.2.4.1.2



Question from College Student Inbox | X

Matt O to ghielec show details Apr 21 (5 days ago) Reply

Hello I am a Computer Engineering student at the University of Central Florida in the United States. I am writing you to ask permission to reprint some figures on your website on my team's Senior Design Document. Your source will be given credit in the works cited of our documentation. Thank you for your time.

http://www.ghielectronics.com/product/2

Reply Forward

from **GHI Electronics Support** <ghielec@ghielectronics.com> hide details Apr 21 (4 days ago) Reply
to Matt O <mattomorrow@gmail.com>
date Tue, Apr 21, 2009 at 4:59 PM
subject Re: Question from College Student

Hello,

That should be fine.

Regards,

GHI Support Team
GHI Electronics, LLC
35555 Garfield Rd. Suite #2
Clinton township, MI 48035
USA
PH: +1 586 693 2696
FAX: +1 586 693 3449
WEB: www.ghielectronics.com
- Show quoted text -

Figure 4.2.4.2.1

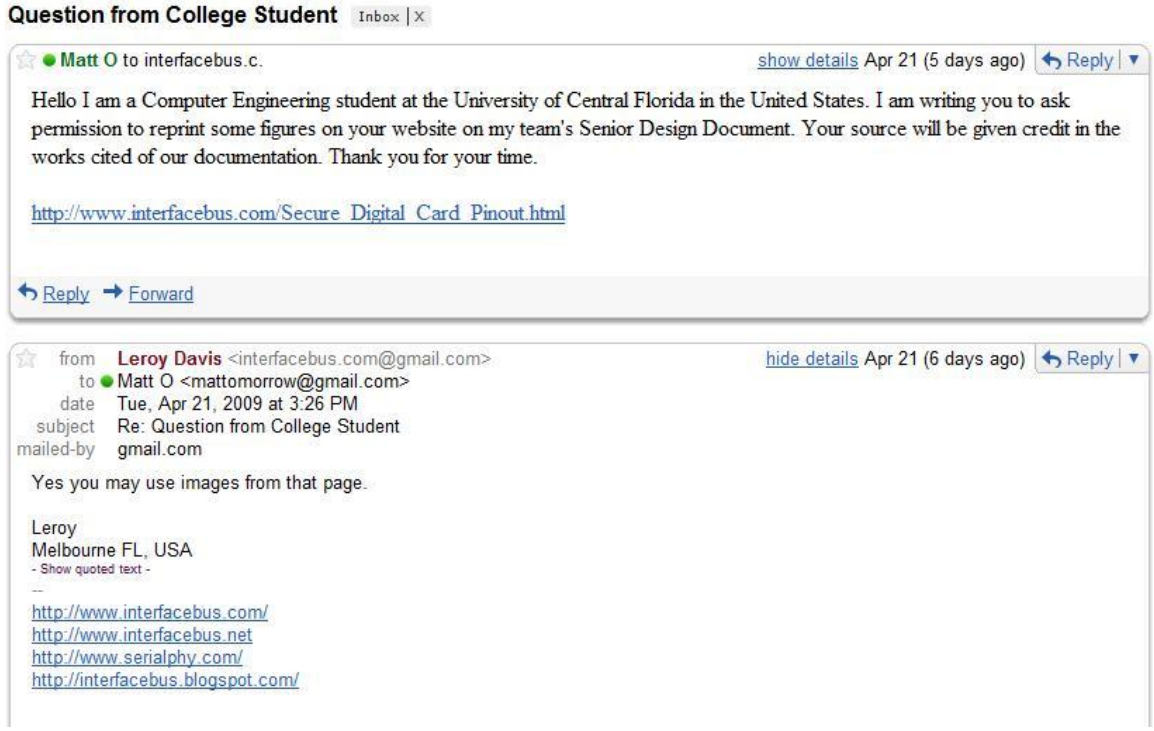

Question from College Student
Inbox | x
Matt O to interfacebus.c.
show details Apr 21 (5 days ago)
Reply
Hello I am a Computer Engineering student at the University of Central Florida in the United States. I am writing you to ask permission to reprint some figures on your website on my team's Senior Design Document. Your source will be given credit in the works cited of our documentation. Thank you for your time.
http://www.interfacebus.com/Secure_Digital_Card_Pinout.html
Reply
Forward
from Leroy Davis <interfacebus.com@gmail.com>
hide details Apr 21 (6 days ago)
Reply
to Matt O <mattomorrow@gmail.com>
date Tue, Apr 21, 2009 at 3:26 PM
subject Re: Question from College Student
mailed-by gmail.com
Yes you may use images from that page.
Leroy
Melbourne FL, USA
- Show quoted text -
--
http://www.interfacebus.com/
http://www.interfacebus.net
http://www.serialphy.com/
http://interfacebus.blogspot.com/

Figure 4.2.4.2.2

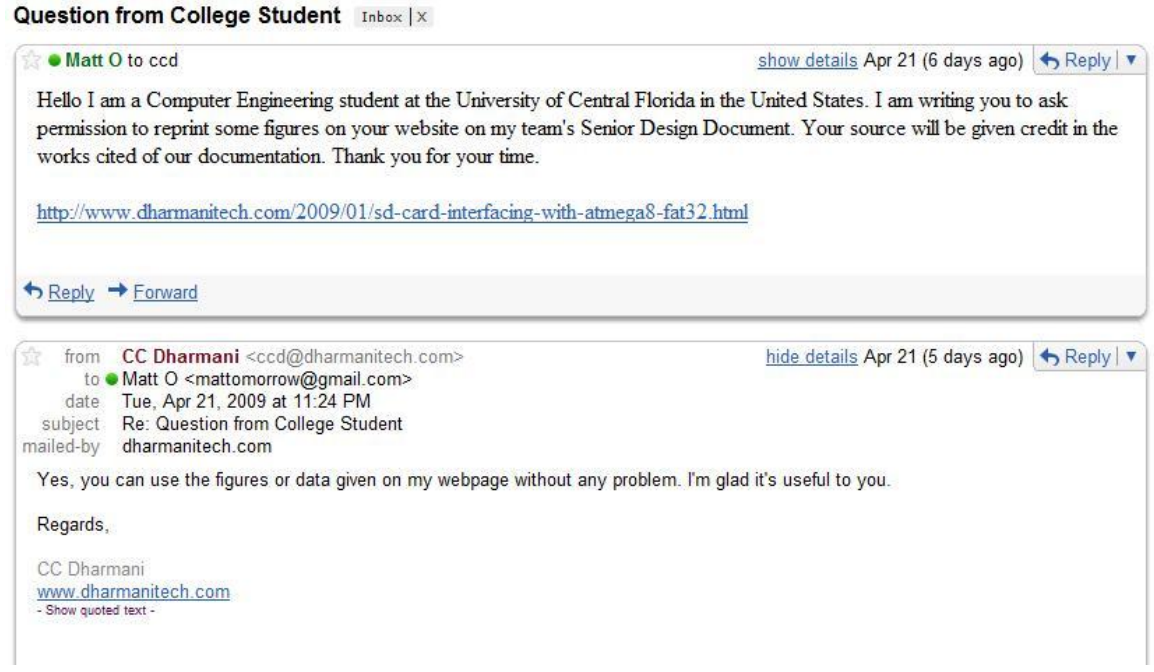
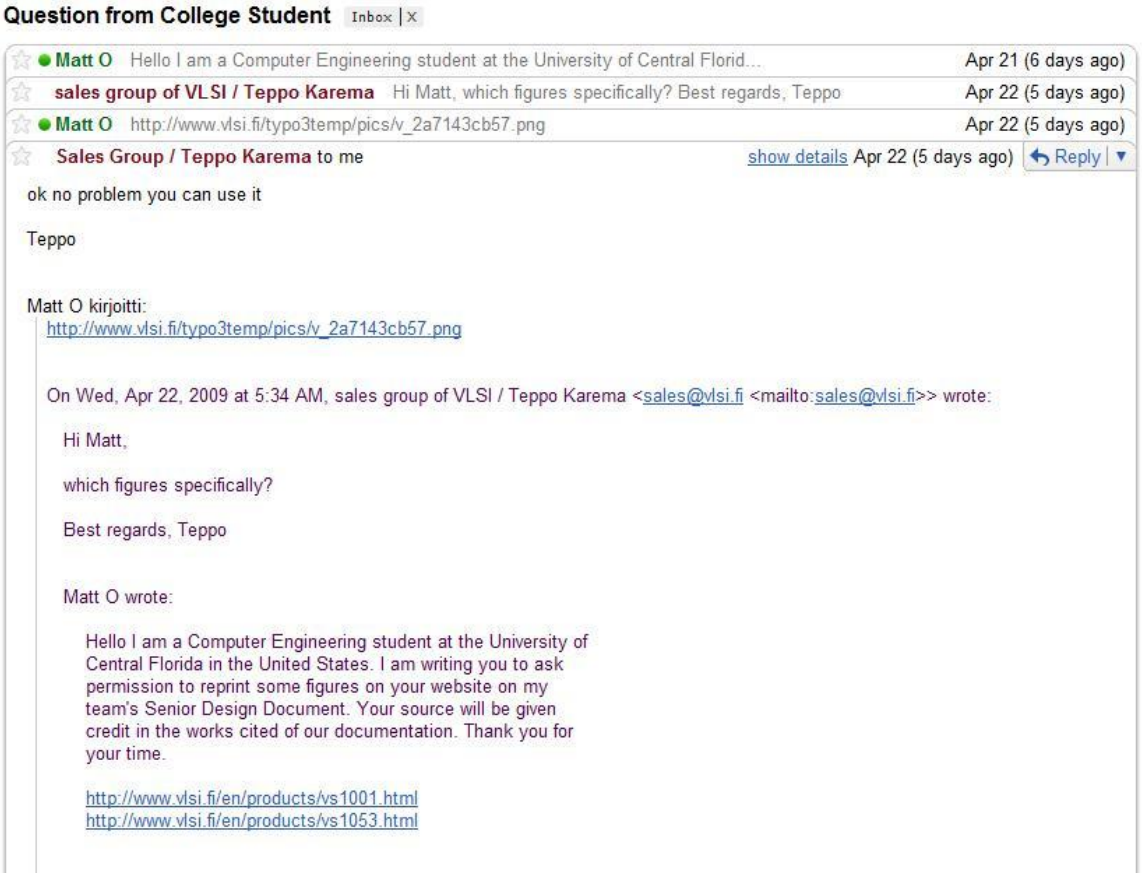

Question from College Student
Inbox | x
Matt O to ccd
show details Apr 21 (6 days ago)
Reply
Hello I am a Computer Engineering student at the University of Central Florida in the United States. I am writing you to ask permission to reprint some figures on your website on my team's Senior Design Document. Your source will be given credit in the works cited of our documentation. Thank you for your time.
http://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html
Reply
Forward
from CC Dharmani <ccd@dharmanitech.com>
hide details Apr 21 (5 days ago)
Reply
to Matt O <mattomorrow@gmail.com>
date Tue, Apr 21, 2009 at 11:24 PM
subject Re: Question from College Student
mailed-by dharmanitech.com
Yes, you can use the figures or data given on my webpage without any problem. I'm glad it's useful to you.
Regards,
CC Dharmani
www.dharmanitech.com
- Show quoted text -

Figure 4.2.3.1.1


Question from College Student
Inbox | x
Matt O Hello I am a Computer Engineering student at the University of Central Florid... Apr 21 (6 days ago)
sales group of VLSI / Teppo Karema Hi Matt, which figures specifically? Best regards, Teppo Apr 22 (5 days ago)
Matt O http://www.vlsi.fi/typo3temp/pics/v_2a7143cb57.png Apr 22 (5 days ago)
Sales Group / Teppo Karema to me show details Apr 22 (5 days ago) Reply
ok no problem you can use it
Teppo
Matt O kirjoitti:
http://www.vlsi.fi/typo3temp/pics/v_2a7143cb57.png
On Wed, Apr 22, 2009 at 5:34 AM, sales group of VLSI / Teppo Karema <sales@vlsi.fi <mailto:sales@vlsi.fi>> wrote:
Hi Matt,
which figures specifically?
Best regards, Teppo
Matt O wrote:
Hello I am a Computer Engineering student at the University of Central Florida in the United States. I am writing you to ask permission to reprint some figures on your website on my team's Senior Design Document. Your source will be given credit in the works cited of our documentation. Thank you for your time.
http://www.vlsi.fi/en/products/vs1001.html
http://www.vlsi.fi/en/products/vs1053.html

Figures 4.2.3.1.2, 4.2.3.2.2

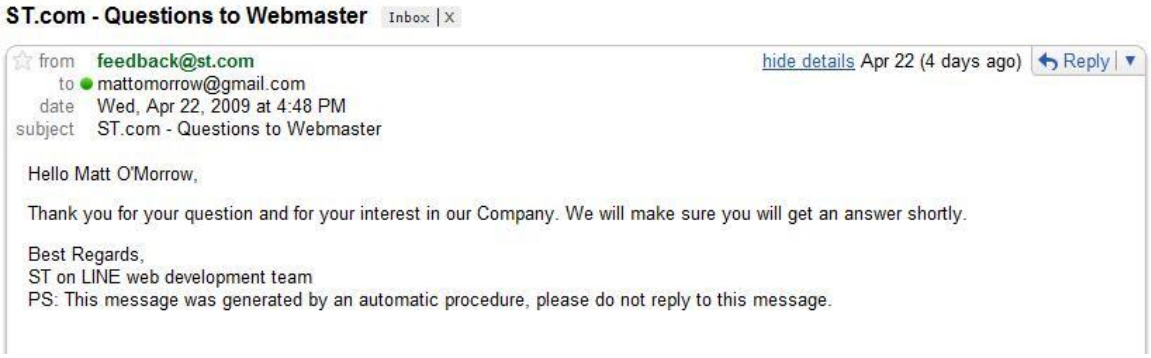

ST.com - Questions to Webmaster
Inbox | x
from feedback@st.com hide details Apr 22 (4 days ago) Reply
to mattomorrow@gmail.com
date Wed, Apr 22, 2009 at 4:48 PM
subject ST.com - Questions to Webmaster
Hello Matt O'Morrow,
Thank you for your question and for your interest in our Company. We will make sure you will get an answer shortly.
Best Regards,
ST on LINE web development team
PS: This message was generated by an automatic procedure, please do not reply to this message.

Figure 4.2.3.2.1

Question from College Student Inbox | X

Matt O to paul show details Apr 21 (6 days ago) Reply

Hello I am a Computer Engineering student at the University of Central Florida in the United States. I am writing you to ask permission to reprint some figures on your website on my team's Senior Design Document. Your source will be given credit in the works cited of our documentation. Thank you for your time.

http://www.pjrc.com/mp3/sta013.html

Reply Forward

Paul Stoffregen to me show details Apr 21 (6 days ago) Reply

Yes, that is fine.

Glad you found the web page useful :-)

- Show quoted text -

Figure 4.2.2.1.1

RE: permission to use superheterodyne block diagram

From: **Steve** (steve@microwaves101.com)
Sent: Mon 3/02/09 8:52 AM
To: 'Andrew Leger' (ajleger85@hotmail.com)

No problem!

Maybe we should add a link to your school here?

http://microwaves101.com/encyclopedia/colleges.cfm

Steve
The Unknown Editor

From: Andrew Leger [mailto:ajleger85@hotmail.com]
Sent: Sunday, March 01, 2009 5:24 PM
To: ue@microwaves101.com
Subject: permission to use superheterodyne block diagram

Hi Unknown Editor,
My name is Andrew Leger, and I am taking Senior Design at the University of Central Florida right now, working towards an undergraduate degree in Electrical Engineering. My group is doing a project that involves FM receiving, and I was hoping to have your permission to use your block diagram of the superheterodyne receiver in our group report (the diagram in question is found at http://www.microwaves101.com/encyclopedia/images/Receivers/block%20diagram.jpg). Of course, proper credit will be given to the website. Thanks for your time, and your response!

Andrew

Figure 4.2.2.1.5

RE: permission to use copyrighted picture

From: **Shannon L. Pleasant** (Shannon.Pleasant@silabs.com)
Sent: Tue 3/03/09 11:36 AM
To: ajleger85@hotmail.com

Hi Andrew, you have permissino to use the image if you could just provide us with the context it will be used in - ie the chapter or section that describes the image. Thanks! Shannon Pleasant, Director of Corporate Communications, Silicon Laboratories

---

From: Andrew Leger
To: DL.Feedback
Sent: Mon Mar 02 18:33:58 2009
Subject: permission to use copyrighted picture

Hi,
My name is Andrew Leger, and I am currently a student at UCF, working towards my undergraduate degree. We are doing a Senior Design project that incorporates an FM tuner, and we were hoping to include the image of the Si4704/05 block diagram (located at https://www.silabs.com/SiteCollectionImages/Block%20Diagrams/Si4704_05_BD.jpg) in our report. Of course, proper credit will be given. Please let me know if that would be acceptable, and thanks in advance. I realize this may not be the proper place to send it, but the 'contact us' section did not have an applicable area.

Thanks,
Andrew

Figures 4.2.4.1.3, 7.1.7.1, 9.2.1.1.2, 9.2.1.1.3, 9.2.1.1.4

From: **Marketing** (marketing@sparkfun.com)
You may not know this sender. Mark as safe | Mark as junk
Sent: Tue 3/10/09 5:08 PM
To: ajleger85@hotmail.com

Hello Andrew. Thanks for asking permission to use our images (and for the proper credit too). Of course you may use any image you like off our website for educational purposes. That goes for anyone else in your class that needs to use our images as well, so please pass along the word.

Good luck on your project!

AnnDrea Boe

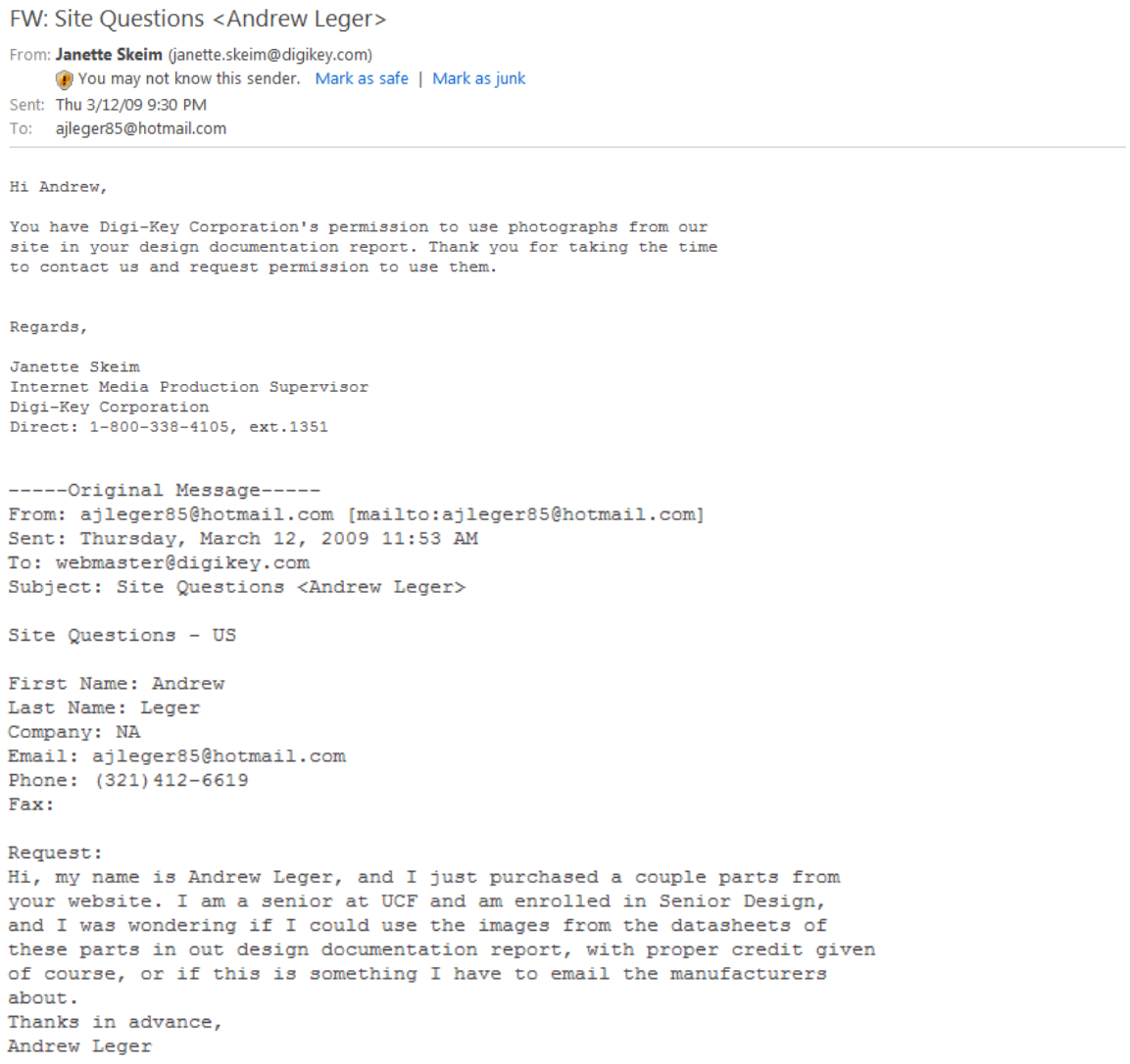
—
Director of Marketing Communications
SparkFun Electronics
6175 Longbow Drive, Suite 200
Boulder, CO 80301

Begin forwarded message:

From: Andrew Leger <ajleger85@hotmail.com>
Date: March 10, 2009 3:00:03 PM MDT
To: <website@sparkfun.com>
Subject: use of copyrighted images

Hi,
My name is Andrew Leger, and I am currently a senior at UCF majoring in electrical engineering. Our senior design project will need a power supply to power a microcontroller among other things, and I was hoping I could use a few of the pictures on http://www.sparkfun.com/commerce/tutorial_info.php?tutorials_id=57 to illustrate some of the basics in our documentation report. Of course, proper credit will be given to the website. Thanks in advance,
Andrew

Figures 4.2.1.4, 4.2.2.1.4, 9.2.1.1.5



FW: Site Questions <Andrew Leger>

From: **Janette Skeim** (janette.skeim@digikey.com)
You may not know this sender. Mark as safe | Mark as junk
Sent: Thu 3/12/09 9:30 PM
To: ajleger85@hotmail.com

Hi Andrew,

You have Digi-Key Corporation's permission to use photographs from our site in your design documentation report. Thank you for taking the time to contact us and request permission to use them.

Regards,

Janette Skeim
Internet Media Production Supervisor
Digi-Key Corporation
Direct: 1-800-338-4105, ext.1351

-----Original Message-----
From: ajleger85@hotmail.com [mailto:ajleger85@hotmail.com]
Sent: Thursday, March 12, 2009 11:53 AM
To: webmaster@digikey.com
Subject: Site Questions <Andrew Leger>

Site Questions - US

First Name: Andrew
Last Name: Leger
Company: NA
Email: ajleger85@hotmail.com
Phone: (321)412-6619
Fax:

Request:
Hi, my name is Andrew Leger, and I just purchased a couple parts from your website. I am a senior at UCF and am enrolled in Senior Design, and I was wondering if I could use the images from the datasheets of these parts in out design documentation report, with proper credit given of course, or if this is something I have to email the manufacturers about.
Thanks in advance,
Andrew Leger

Figure 4.2.2.1.2

superheterodyne receiver schematic

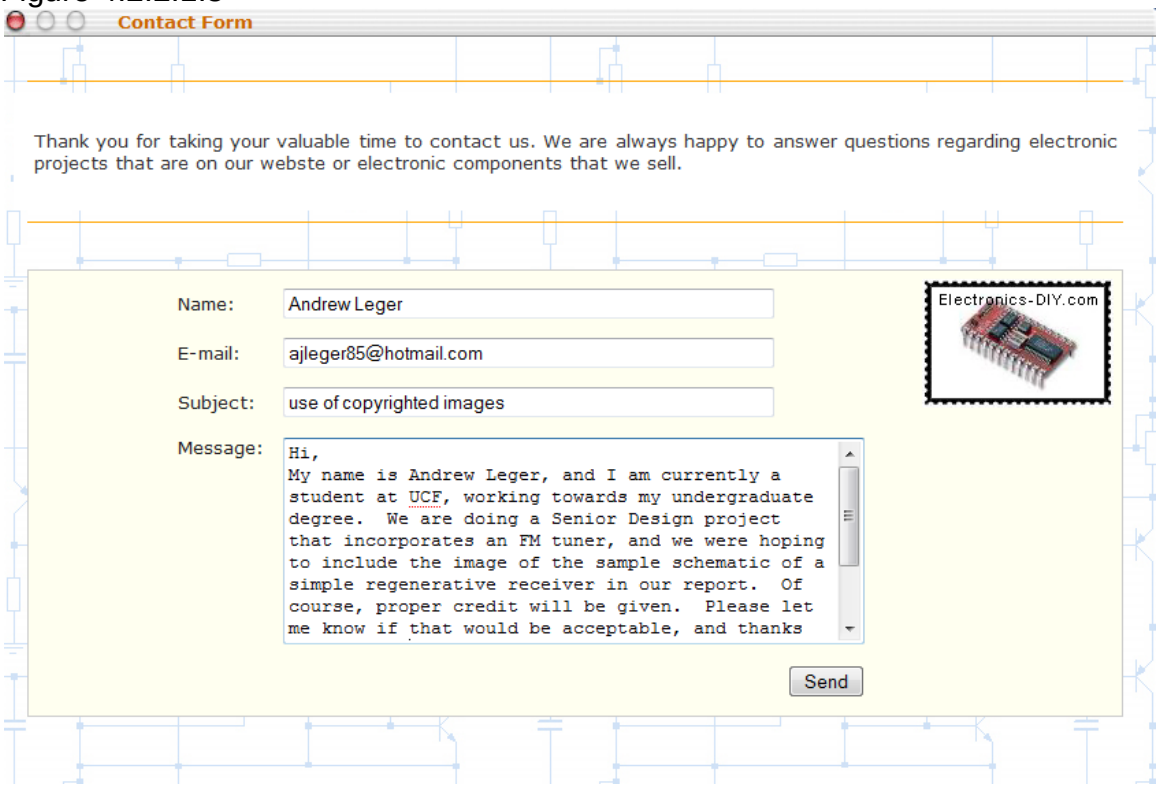
From: **Andrew Leger** (ajleger85@hotmail.com)
Sent: Mon 3/02/09 10:28 AM
To: webmaster@hans-egebo.dk

Hi,

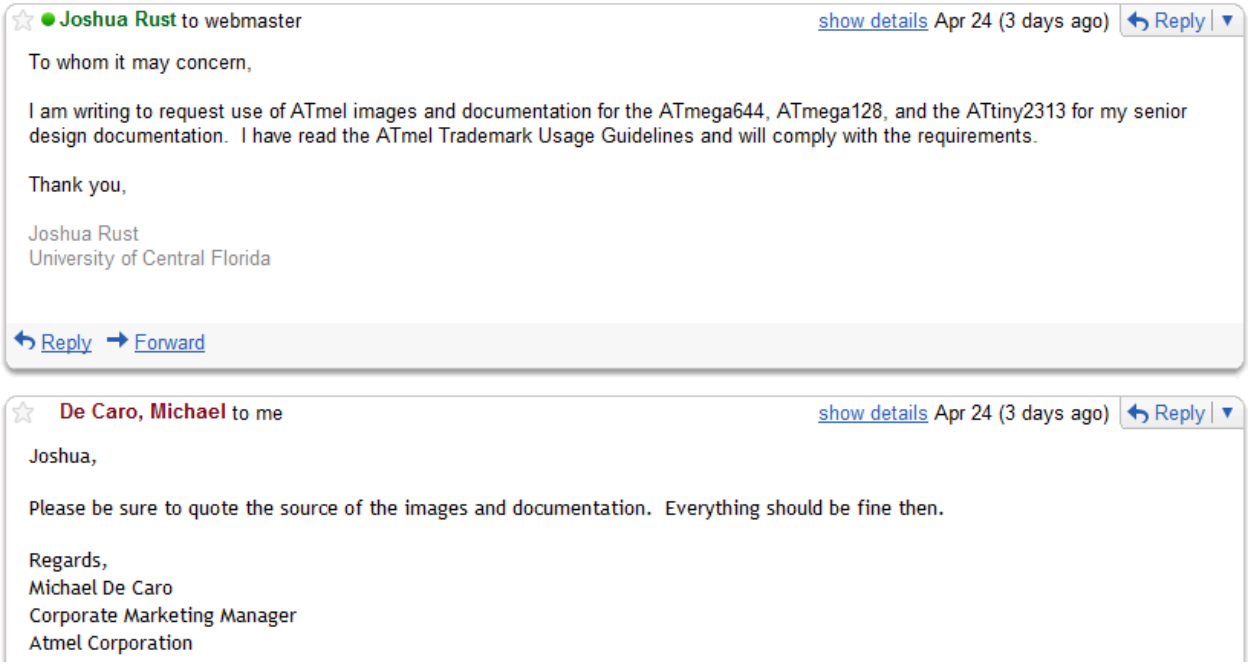
My name is Andrew Leger, and I am a student at the University of Central Florida majoring in Electrical Engineering. My senior design group is doing a project that involves FM reception, and we were hoping to include your schematic of a superheterodyne receiver (located at http://www.hans-egebo.dk/Tutorial/images/Superhet.gif) in our documentation report. Of course, proper credit will be given. Thanks in advance,

Andrew

Figure 4.2.2.1.3


Contact Form
Thank you for taking your valuable time to contact us. We are always happy to answer questions regarding electronic projects that are on our webste or electronic components that we sell.
Name:
Andrew Leger
E-mail:
ajleger85@hotmail.com
Subject:
use of copyrighted images
Message:
Hi,
My name is Andrew Leger, and I am currently a student at UCF, working towards my undergraduate degree. We are doing a Senior Design project that incorporates an FM tuner, and we were hoping to include the image of the sample schematic of a simple regenerative receiver in our report. Of course, proper credit will be given. Please let me know if that would be acceptable, and thanks
Send
Electronics-DIY.com

Figurew 3.2.2.2.1, **Figure 3.3.1, Table 7.2.2.2.1**



Joshua Rust to webmaster show details Apr 24 (3 days ago) Reply

To whom it may concern,

I am writing to request use of ATmel images and documentation for the ATmega644, ATmega128, and the ATtiny2313 for my senior design documentation. I have read the ATmel Trademark Usage Guidelines and will comply with the requirements.

Thank you,

Joshua Rust
University of Central Florida

Reply Forward

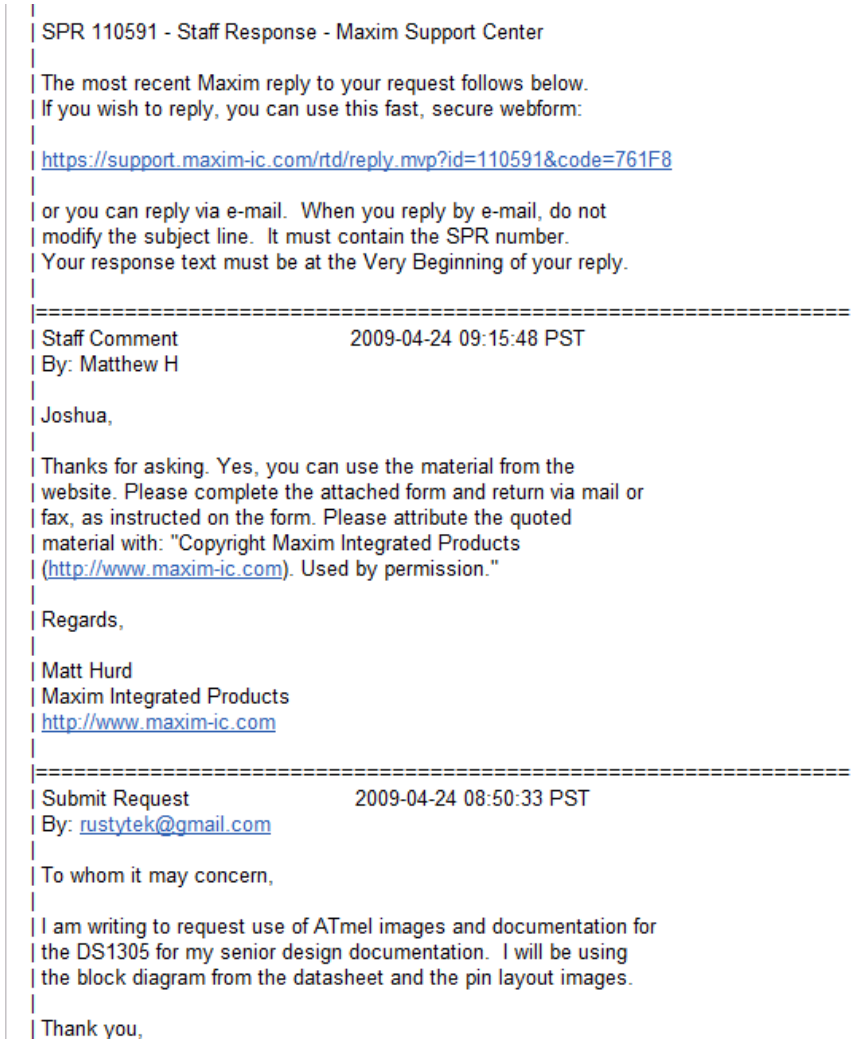
De Caro, Michael to me show details Apr 24 (3 days ago) Reply

Joshua,

Please be sure to quote the source of the images and documentation. Everything should be fine then.

Regards,
Michael De Caro
Corporate Marketing Manager
Atmel Corporation

Figures **Figure 8.2.2.1, Figure 8.2.2.2**



| SPR 110591 - Staff Response - Maxim Support Center
|
| The most recent Maxim reply to your request follows below.
| If you wish to reply, you can use this fast, secure webform:
|
| https://support.maxim-ic.com/rtd/reply.mvp?id=110591&code=761F8
|
| or you can reply via e-mail. When you reply by e-mail, do not
| modify the subject line. It must contain the SPR number.
| Your response text must be at the Very Beginning of your reply.
|
|==========
| Staff Comment 2009-04-24 09:15:48 PST
| By: Matthew H
|
| Joshua,
|
| Thanks for asking. Yes, you can use the material from the
| website. Please complete the attached form and return via mail or
| fax, as instructed on the form. Please attribute the quoted
| material with: "Copyright Maxim Integrated Products
| (http://www.maxim-ic.com). Used by permission."
|
| Regards,
|
| Matt Hurd
| Maxim Integrated Products
| http://www.maxim-ic.com
|
|==========
| Submit Request 2009-04-24 08:50:33 PST
| By: rustytek@gmail.com
|
| To whom it may concern,
|
| I am writing to request use of ATmel images and documentation for
| the DS1305 for my senior design documentation. I will be using
| the block diagram from the datasheet and the pin layout images.
|
| Thank you,

FIGURE 6.1.1.1.1 and FIGURE 6.1.1.2.1

From: Navid Mokhberi <nmokhberi@futek.com>
Date: March 23, 2009 3:15:15 PM EDT
To: <koyapb@earthlink.net>
Subject: Re: website image inquiry

Phillip,

Thanks for the response and explanation of usage. You have our permission for usage. Please keep this email for your reference.

Thanks

Navid Mokhberi
Business Development Associate
navid@futek.com

FUTEK Advanced Sensor Technology, Inc.
10 Thomas, Irvine, CA 92618
V: 949.297.9660
F: 949.465.0905
Toll: 800.23.FUTEK
www.futek.com

On 3/21/09 2:42 PM, "koyapb@earthlink.net" <koyapb@earthlink.net> wrote:

> Thanks for your quick reply
>
> Me and my teammates are researching options for the design of a system
> that needs to detect if a person is in a bed, and hopefully be able to
> tell the difference between a person and a pet or luggage. So I'm

> just looking into what's out there as far as pressure/weight sensing
> goes, though we most likely will just use infrared, pressure sensing
> would have been useful for measuring exactly how much the overall
> weight including the mattress would be, allowing the system to be
> further programmed to handle multiple bed users at once, that is if
> the system knows how much each person weighs, it can tell who has left
> the bed and who hasn't.
>
> I didn't have an exact product in mind, though I was going to make use
> of the base product images for the pancake, load button, and donut
> load cells
>
> http://www.futek.com/product.aspx?stock=FSH00126&acc2=acc
> http://www.futek.com/product.aspx?stock=FSH00158&acc2=acc
> http://www.futek.com/product.aspx?stock=FSH00294&acc2=acc
>
> especially the "in use" sketch for the donut load cell to demonstrate
> how they could be positioned with a rod through the center attached to
> a board that is sized to fit under the entire mattress, allowing for
> better weight measuring.
>
> also an image demonstrating how thin the force sensors are such as
>
> http://www.futek.com/product.aspx?t=force&m=ffp350
>
> of course most of these are more industrial testing and due to

the
> complications of setup for our particular project will probably not be
> used, but our professor encourages fully researching any options.
>
> Thank you
>
> Philip Bell
> koyapb@earthlink.net
>
>
> On Mar 21, 2009, at 5:20 PM, Navid Mokhberi wrote:
>
>> Hi Philip,
>>
>> Thanks for emailing us first. We're naturally cautious how our

>> product images are used. Our general position is to allow usage as
>> long as the document/research paper doesn't position FUTEK in a
>> negative manner. Which product did you have in mind and what is the
>> synopsis of your research paper?
>>
>> Regards,
>>
>> Navid Mokhberi
>> Business Development Associate
>> navid@futek.com
>>
>> FUTEK Advanced Sensor Technology, Inc.
>> 10 Thomas, Irvine, CA 92618
>> V: 949.297.9660
>> F: 949.465.0905
>> Toll: 800.23.FUTEK
>> www.futek.com
>>
>>
>> -----Original Message-----

>> From: koyapb@earthlink.net [mailto:koyapb@earthlink.net]
>> Sent: Sat 3/21/2009 10:14 AM
>> To: Navid Mokhberi
>> Subject: website image inquiry
>>
>> Hello
>>
>> My name is Philip Bell
>> I am a Computer Engineering student at the Universitty of Central
>> Florida in Orlando Florida
>>
>> I am emailing to request permission to use some of the product images
>> for a senior design project documentation. Please let me know any
>> additional information you need from me. Thank you
>>
>> Philip Bell
>> koyapb@earthlink,net
>>
>

Figure 4.2.3.2.1:

From: Paul Stoffregen <paul@pjrc.com>
Date: March 24, 2009 11:20:35 PM EDT
To: koyapb@earthlink.net
Subject: Re: [Fwd: website image inquiry]

Did you have any specific image in mind?

In general, we don't mind if you use material from the pjrc.com website for educational (non-profit) purposes.

Of course, if you require specific affirmative permission to use an image, you need to be specific about which image and how you're using it.

Robin Coon wrote:

Subject: website image inquiry
From: koyapb@earthlink.net
Date: Sat, 21 Mar 2009 13:01:05 -0400
To: robin@pjrc.com
To: robin@pjrc.com

Hello

My name is Philip Bell
I am a Computer Engineering student at the Universitty of Central Florida in Orlando, Florida

I am emailing to request permission to use some of the product images for a senior design project documentation. Please let me know any additional information you need from me. Thank you

Philip Bell
koyapb@earthlink,net

FIGURE 5.3.2.1

From: "sales" <sales@futurlec.com>
Date: April 12, 2009 6:16:43 PM EDT
To: <koyapb@ix.netcom.com>
Subject: RE: Request for use of images from the website

Dear Sir,

Thanks for your enquiry and yes, it is fine to use these images in your design documents.

Best Regards
Alan
Sales Manager
Futurlec
http://www.futurlec.com

From: koyapb@ix.netcom.com [mailto:koyapb@ix.netcom.com]
Sent: Sunday, 12 April 2009 10:44 PM
To: technical@futurlec.com
Cc: webmaster@futurlec.com; sales@futurlec.com
Subject: Request for use of images from the website

Hello, I am re-emailing a previous email that failed to deliver

My name is Philip Bell
I am a college student at University of Central Florida and wanted to request permission to use some product images in a design document, here is the email I sent previously and the message the server sent back. Let me know if there is anything else you need from me. Thank you

==========
PREVIOUS MESSAGE
==========

This message was created automatically by mail delivery software.

A message that you sent could not be delivered to one or more of its

recipients. This is a permanent error. The following address(es) failed:

save to /clientmail/mail-1/f/u/futurlec.com/webmaster/
generated by webmaster@futurlec.com
mailbox is full: retry timeout exceeded

------ This is a copy of the message, including all the headers. ------

Return-path: <koyapb@earthlink.net>
Received: from pop06.mail.atl.earthlink.net ([207.69.200.40])
by smtp-mx-server-8.servers.netregistry.net protocol: esmtp (Exim 4.69 #1 (Debian))
id 1Ll4Y6-0005tq-59
for <webmaster@futurlec.com>; Sun, 22 Mar 2009 03:59:54 +1100
Received: from user-142ga72.cable.mindspring.com ([72.40.40.226] helo=[10.0.1.199])
by pop06.mail.atl.earthlink.net with esmtp (Exim 3.36 #1)
id 1Ll4YB-0005zT-00
for webmaster@futurlec.com; Sat, 21 Mar 2009 12:59:59 -0400
Message-Id: <150A126A-6B26-4EEF-8020-537DF4156B13@earthlink.net>
From: koyapb@earthlink.net
To: webmaster@futurlec.com
Content-Type: text/plain; charset=US-ASCII; format=flowed; delsp=yes
Content-Transfer-Encoding: 7bit
Mime-Version: 1.0 (Apple Message framework v930.3)
Subject: Use of website images
Date: Sat, 21 Mar 2009 12:59:59 -0400
X-Mailer: Apple Mail (2.930.3)

Hello

My name is Philip Bell
I am a Computer Engineering student at the Universitty of Central Florida in Orlando Florida

I am emailing to request permission to use some of the product images for a senior design project documentation. Please let me know any additional information you need from me. Thank you

Philip Bell
koyapb@earthlink,net

Figure 6.1.1.3.1

From: "Amy McDonough" <amcdonough@tekscan.com>
Date: March 23, 2009 4:08:12 PM EDT
To: "Philip Bell" <koyapb@earthlink.net>
Subject: FlexiForce Images

Dear Philip,

Thank you for emailing to request the use of our images. You are welcome to use any images on the website. Please let me know if you need anything else.

Best regards,
Amy

Amy McDonough *FlexiForce Sales & Marketing Assistant*
Tekscan, Inc. Phone: 617.464.4500 x245
amcdonough@tekscan.com www.tekscan.com **Order FlexiForce sensors and ELF Systems at our online shopping cart**